SCENE ANALYSIS USING RECURSIVE FREQUENCY DOMAIN CORRELATION WITH ENERGY NORMALIZATION

THESIS

Richard L. Mills
Second Lieutenant, USAF

AFIT/GEO/ENG/84D-3

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY
AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

SCENE ANALYSIS USING RECURSIVE FREQUENCY DOMAIN CORRELATION WITH ENERGY NORMALIZATION

THESIS

Richard L. Mills
Second Lieutenant, USAF

AFIT/GEO/ENG/84D-3

AFIT/GEO/ENG/84D-3

SCENE ANALYSIS USING RECURSIVE FREQUENCY DOMAIN CORRELATION WITH ENERGY NORMALIZATION

THESIS

Presented to the Faculty of the School of Engineering

of the Air Force Institute of Technology

Air University

In Partial Fulfillment of the

Requirements for the Degree of

Master of Science in Electrical Engineering

Richard L. Mills, B.S.

Second Lieutenant, USAF

December 1984

Acknowledgements

"Seek first the kingdom of God and His righteousness, and all these things will be added to you." Matt 6:33.

I praise God for the ability and the opportunity to do this thesis. I want to thank my thesis advisor, Dr. Matthew Kabrisky, for his ideas and insight on this problem. I thank Dan Zambon, the system manager, for his efforts in keeping the Signal Processing Laboratory in working order when circumstances made it seem impossible to do so. I also want to thank Capt. David King for his expertise in trouble-shooting my software problems.

I give special thanks to my wife, Elizabeth, who gave me unlimited support, not only during this thesis study, but also throughout my entire stay at AFIT.

Contents

		Page
Preface		ii
List of Figures		v
List of Tables		viii
Abstract		ix
I.	Introduction	1
1.1	Background	1
1.2	Problem	2
1.3	Scope	3
1.4	Assumptions	4
1.5	Approach	6
1.6	Equipment	6
II.	Preprocessing	7
2.1	Video Files	7
2.2	Edge Detection Algorithm	7
III.	First Pass Correlation Process	17
3.1	Template Generation	17
3.2	Two Dimensional Fast Fourier Transform	23
3.3	File Multiplication	24
3.4	Inverse Two Dimensional Fast Fourier Transform	24
3.5	Output	25
3.6	Fourier Filtering	27
IV.	Second Pass Correlation Process	36
4.1	File Regeneration	36
4.2	Energy Normalization	41
4.3	Recorrelation	42
V.	Evaluation	47
5.1	Three Dimensional Representation	47
5.2	Histogram Analysis	48
5.3	Evaluation Process	55
5.4	Results	59
VI.	Discussion	61
6.1	Conclusions	61
6.2	Recommendations	62
6.3	Process Summary	62

Bibliography 65

Appendix A: Materials and Equipment 66

Appendix B: Preprocessing Software 74

Appendix C: Main Software 83

Appendix D: Support Software 117

Appendix E: Test Cases 128

Test Case II 129
Test Case III 133
Test Case IV 141
Test Case V 151
Test Case VI 153
Test Case VII 166

Vita 173

List of Figures

Figure		Page
2.1	Masks of the Wedge Operator	9
2.2	256x256 Pixel Representation of F-16's	10
2.3	Edge Extraction in the Horizontal Direction	11
2.4	Edge Extraction in the First Diagonal Direction	12
2.5	Edge Extraction in the Vertical Direction	13
2.6	Edge Extraction in the Second Diagonal Direction	14
2.7	Combination if the Files of Figures 2.3-2.6 (Complete Edge Extraction)	16
3.1	Template of F-16 Extracted from Figure 2.7	19
3.2	OCTEK Cursor Movement (Function #4)	20
3.3	Image Information Movement Due to SPIN	21
3.4	Final Video File of Template of F-16	22
3.5	Correlation Output of Target Scene with Origin-Centered Template of F-16	26
3.6	Display Monitor Output of Figure 3.5	27
3.7	Wedged F-16 Templates	29
3.8	Correlation Output of Target Scene with Gereral F-16 Template	30
3.9	Display Monitor Output of Figure 3.8	31
3.10	Correlation Output of Target Scene with General F-16 Template using 7x7 Pixel Filter	33
3.11	Correlation Output of Target Scene with General F-16 Template using 39x39 Pixel Filter	34
3.12	Display Monitor Output of Figure 3.11	35
4.1	Small Files Before and After VSPIN	36
4.2	Small Files Corresponding to Peaks	40
4.3	Some Small Files Before and After NORM	44

4.4 Second Pass Correlation Output (a-c) 45

4.4 Second Pass Correlation Output (d-e) 46

5.1 Three-Dimensional Representation of Auto-
correlation 49

5.2 Three-Dimensional Representation of Figure 4.4b . 50

5.3 Three-Dimensional Representation of Figure 4.4e . 51

5.4 Histogram of Pixel Population of Figure 4.4a . . . 52

5.5 Histogram of Pixel Population of Figure 4.4b . . . 53

5.6 Histogram of Pixel Population of Figure 4.4e . . . 54

5.7 Function of the Scorel Evaluation 56

5.8 Second Pass Correlation Output Files, With and
Without Filtering 60

6.1a Flowchart of Process (Part I) 63

6.1b Flowchart of Process (Part II) 64

E.1a-c Second Pass Correlation Output of Case II . . .130

E.1d-e Second Pass Correlation Output of Case II . . .131

E.2 Input Image of Test Case III 134

E.3 Edge Extraction of Figure E.2 135

E.4 T-38 Templates .136

E.5 First Pass Correlation Output of Test Case III . .137

E.6 Small Files Corresponding to Peaks in Figure E.5 .138

E.7 Second Pass Correlation Output of Test Case III .139

E.8 Input Image of Test Case IV 142

E.9 Edge Extraction of Figure E.8 143

E.10 F-86 Template .144

E.11 First Pass Correlation Output of Test Case IV . .145

E.12a-d Small Files Corresponding to Peaks in
Figure E.11 146

E.12e-h Small Files Corresponding it Peaks in Figure E.11 147

E.13a-d Second Pass Correlation Output of Test Case IV 148

E.13e-h Second Pass Correlation Output of Test Case IV 149

E.14 Input Image of Test Case V 152

E.15 Edge Extraction of Figure E.14 153

E.16 B-52 Template .154

E.17 First Pass Correlation Output of Test Case V 155

E.18 Small Files Corresponding to Peaks in Figure E.17 156

E.19 Second Pass Correlation Output of Test Case V . . .157

E.20 Input Image of Test Case VI 160

E.21 Edge Extraction of Figure E.20 161

E.22 First Pass Correlation Output of Test Case VI . . .162

E.23 Small Files Corresponding to Peaks in Figure E.22 163

E.24 Second Pass Correlation Output of Test Case VI . . .164

E.25 Input Image of Test Case VII 167

E.26 Edge Extraction of Figure E.25 168

E.27 First Pass Correlation Output of Test Case VII . . .169

E.28 Small Files Corresponding to Peaks in Figure E.27 170

E.29 Second Pass Correlation Output of Test Case VII. . .171

List of Tables

Table		Page
I	Evaluation of Second Pass Correlation Output Files	58
II	Evaluation of Test Case II	132
III	Evaluation of Test Case III	140
IV	Evaluation of Test Case IV	150
V	Evaluation of Test Case V	158
VI	Evaluation of Test Case VI	165
VII	Evaluation of Test Case VII	172

ABSTRACT

This thesis describes a scene analysis algorithm which locates targets in a noisy background. Preprocessing is used to detect the edges of objects in a digitized scene. The edges extracted from the scene is then cross-correlated with a template of the target to be found.

Cross-correlation is done by using complex-conjugate multiplication in the frequency domain. Areas of high correlation are recorrelated using smaller images which can be processed faster. The potential targets are energy normalized with respect to the template in order to eliminate false correlation with noise.

Even in a scene with high energy noise, the algorithm works very well when operated under the constraints of size, orientation, and perspective. originator-supplied keywords:

→ to 1473, field 19

SCENE ANALYSIS

USING RECURSIVE FREQUENCY DOMAIN CORRELATION

WITH ENERGY NORMALIZATION

I. INTRODUCTION

1.1 Background

Image pattern recognition is concerned with extracting discriminating features from visual data and using these features to identify objects in the image. The human mind can perform this function with little trouble; however, machines cannot. The applications of image pattern recognition lie primarily in four major areas: (Ref 8)

1. Document processing
2. Industrial automation
3. Medicine and biology
4. Military target finding/guidance.

This thesis deals mainly with the military applications of target recognition; however, the results presented in this thesis can be extended to the other areas.

With recent advances in visual and infrared detector technology, the problem of collecting image data is a relatively minor obstacle. The main problem is the interpretation of an image: filtering out the unwanted visual data to determine if a target is present.

Although there has been much work done in this area, no perfect algorithm has yet been produced which can process the visual data collected. One reason is because the level of the scene clutter (noise energy) varies for every scene. This thesis deals mainly with the elimination of all noise.

Much work has been done on this area at the AFIT Signal Processing Laboratory. This thesis is a continuation of four recent theses done at AFIT: Horev, Hamadoni, Cromer, and Wells (Ref 3,2,1,10); as well as the proposed dissertation by McKeeman (Ref 7).

1.2 Problem

An image of a scene that contains a potential target has more information that just that from the target. For example, infrared radiation is detected from the targets, as well as from the background of the scene. The computer which handles the visual data must be able to distinguish the target from the background radiation.

The problem of locating targets in a scene has six major variables:

1. Target position
2. Target shape
3. Target size
4. Target perspective
5. Target rotation (orientation)
6. Scene clutter (noise).

The determination of all target positions within a scene is the goal of this thesis. The target shape is the property which characterizes the target. The size and perspective of the target are also variables within the problem, but they may be known in a given scenario (for example, observing tanks from an airplane of known altitude). The main thrust of this study was to locate the targets regardless of the amount of scene clutter (noise).

The standard method for the collection of this visual data is to represent the scene with an array of pixels (picture elements). The standard image file used at the AFIT Signal Processing Lab contains 65,536 pixels (256x256). Each pixel contains 4 bits. This allows for 16 different gray levels (0 to 15) which correspond to the intensity of the radiation at each pixel point.

When the image is taken at a relatively close distance from the scene the targets may be distinguishable to a human observer. When the image is taken at a relatively far distance from the scene the targets appear as blobs. Often times even a human observer cannot identify the blobs as being targets.

The problem is to develop an algorithm which manipulates the image data so that the targets may be located and identified with little or no human intervention.

1.3 Scope

This thesis deals only with the processing of the image data collected from a scene. The method of collection, optics involved, atmospheric transmission, and wavelengths used, are not considered.

The scope of this thesis confines the software to be used on the computers within the AFIT Signal Processing Lab. The algorithms were tested on visual image data created within the lab, although the process is not neccessarily confined to visual data. A scene represented in a two-dimensional array of pixels, visual or infrared, can be operated on by these algorithms.

Constraints on the computer run times were not considered in this thesis. The purpose was only to develop an image processing algorithm which achieved the goal of target identification. The philosophy taken was that if the proper algorithm could be developed in the Signal Processing Lab, regardless of run time, then it may be implemented in a fast running, dedicated VLSI system (Ref 4).

The software written has the flexibility to work with several sizes of video data files. However, due to computer run times with the larger files, many of the secondary steps use only files sizes of 64x64 pixels.

1.4 Assumptions

The algorithms developed perform under the following assumptions concerning the input test scenes:

1. The digitized scene images contained multiple targets which were similar in size, persective, and orientation.

2. The size, perspective and orientation of the targets contained within the scene is known. As stated

earlier, size and perspective may be given in a specific scenario. The problem of orientation may be solved by re-analyzing the scene for each possible orientation of the target. This is believed to be much the same way that the human mind processes images that it sees (Ref 4).

3. The location and relative energy of the targets is not known. The amount of scene clutter is also unknown.

4. When only images of size 64x64 were used for recorrelation it is assumed that the target was no more that 64 pixels in size in any direction or that an area on the target of that size contained the neccessary identifying visual information.

5. The unknown locations of the targets may be anywhere in the scene. However, indentification of targets close to the edge of the scene may be somewhat difficult compared to those which are at least half the distance of the recorrelation file size away form the border.

6. The digitized scene images may contain noise (scene clutter). This noise may be of high or low energy level.

7. The separation of individual targets is great enough to allow for only one target to be contained in each of the smaller images used for recorrelation.

1.5 Approach

The process of target recognition described in this thesis is divided into four major parts:

1. Preprocessing of the scene to eliminate background noise and enhance the target edges. The edges of the objects are used to constrain the identifying characteristics of the potential targets and to reduce overall scene clutter. Edge detection is also used because it is a form of low-level processing performed by the human visual system.

2. First pass correlation of the edge-detected scene with a template of the target to be located. This correlation is done to identify potential target locations and is performed in the frequency domain.

3. Second pass correlation (in the frequency domain) of potential target locations with the target template. Energy normalization is performed on the potential target to enhance correlation with true targets and attenuate correlation with false targets.

4. Evaluation of the recorrelation video files.

1.6 Equipment

All equipment used in this thesis resides in the AFIT Signal Processing Laboratory. All software written remains on magnetic tape at the lab. Appendix A contains a brief description of the equipment and software support available in the lab.

II. PREPOCESSING

2.1 Video Files

Most of the software in this thesis is designed to operate on video files of the size 256x256 pixels. Each pixel contains an integer value corresponding to the intensity of the radiation at that particular point in the scene. These values range from 0, for total black, to 15, for total white. The software may be modified to handle images of different sizes or of different pixel resolutions.

This pixel resolution requires 4 bits to represent each point. For economy of storage the video files were condensed so that each 16-bit word contains the information of 4 pixels. The video files were configured in this manner with the subroutine REPACK and retrieved with the subroutine UNPACK. The source code for these subroutines is contained in Appendix D.

2.2 Edge Detection Algorithm

Before the targets are extracted from the scene, a preprocessing algorithm is performed to reduce noise. The Wedge function was used to do this by detecting the visible edges in the scene. The Wedge operator was used because it was found to be the most effective and the most consistent of the edge detection operators that have been studied (Ref 7). The Wedge algorithm used was that of Wells's work in 1983 (Ref 10).

This process has two steps. The first is the extraction of edges in four directions. The second step is the combining of the four edge extractions into one video file.

The first step is performed by the program WEDGE. This program accepts the name of a video file and the names of 4 output files to be created. Each of the four output files contains the edges of the scene in a certain direction (vertical, horizontal, and the two diagonals). WEDGE uses several 3x3 masks to detect edges of pixel intensity. This is implemented by the subroutine WROW1, which uses a four element wedge operator (Ref 10). Figure 2.1 shows the masks used by WROW1. Each of the four directions has four masks which are used to detect edges in the given direction.

The program WEDGE examines the value of each pixel and the pixels which surround it. Each of the nine pixel values are multiplied by the corresponding coefficient of the mask. Into one of the four output files is put a value which indicates the level of the greatest degree of sharpness of the edge in the corresponding direction. This process is continued for every pixel point.

Figure 2.2 contains a 256x256 pixel representation of a group of F-16's. Figures 2.3-2.6 contain the four output video files of the four different masking operations.

Direction #1 (Horizontal)

```
+---+---+---+   +---+---+---+   +---+---+---+   +---+---+---+
| 2 | 2 | 2 |   |-1 |-1 |-1 |   |-2 |-2 |-2 |   | 1 | 1 | 1 |
+---+---+---+   +---+---+---+   +---+---+---+   +---+---+---+
|-1 |-1 |-1 |   |-1 |-1 |-1 |   | 1 | 1 | 1 |   | 1 | 1 | 1 |
+---+---+---+   +---+---+---+   +---+---+---+   +---+---+---+
|-1 |-1 |-1 |   | 2 | 2 | 2 |   | 1 | 1 | 1 |   |-2 |-2 |-2 |
+---+---+---+   +---+---+---+   +---+---+---+   +---+---+---+
```

Direction #2 (1st Diagonal)

```
+---+---+---+   +---+---+---+   +---+---+---+   +---+---+---+
|-1 | 2 | 2 |   |-1 |-1 |-1 |   | 1 |-2 |-2 |   | 1 | 1 | 1 |
+---+---+---+   +---+---+---+   +---+---+---+   +---+---+---+
|-1 |-1 | 2 |   | 2 |-1 |-1 |   | 1 | 1 |-2 |   |-2 | 1 | 1 |
+---+---+---+   +---+---+---+   +---+---+---+   +---+---+---+
|-1 |-1 |-1 |   | 2 | 2 |-1 |   | 1 | 1 | 1 |   |-2 |-2 | 1 |
+---+---+---+   +---+---+---+   +---+---+---+   +---+---+---+
```

Direction #3 (Vertical)

```
+---+---+---+   +---+---+---+   +---+---+---+   +---+---+---+
|-1 |-1 | 2 |   | 2 |-1 |-1 |   | 1 | 1 |-2 |   |-2 | 1 | 1 |
+---+---+---+   +---+---+---+   +---+---+---+   +---+---+---+
|-1 |-1 | 2 |   | 2 |-1 |-1 |   | 1 | 1 |-2 |   |-2 | 1 | 1 |
+---+---+---+   +---+---+---+   +---+---+---+   +---+---+---+
|-1 |-1 | 2 |   | 2 |-1 |-1 |   | 1 | 1 |-2 |   |-2 | 1 | 1 |
+---+---+---+   +---+---+---+   +---+---+---+   +---+---+---+
```

Direction #4 (2nd Diagonal)

```
+---+---+---+   +---+---+---+   +---+---+---+   +---+---+---+
|-1 |-1 |-1 |   | 2 | 2 |-1 |   | 1 | 1 | [illegible] |   |-2 |-2 | 1 |
+---+---+---+   +---+---+---+   +---+---+---+   +---+---+---+
|-1 |-1 | 2 |   | 2 |-1 |-1 |   | 1 | 1 |-2 |   |-2 | 1 | 1 |
+---+---+---+   +---+---+---+   +---+---+---+   +---+---+---+
|-1 | 2 | 2 |   |-1 |-1 |-1 |   | 1 |-2 |-2 |   | 1 | 1 | 1 |
+---+---+---+   +---+---+---+   +---+---+---+   +---+---+---+
```

Figure 2.1 Masks of the Wedge Operator (Ref 10).

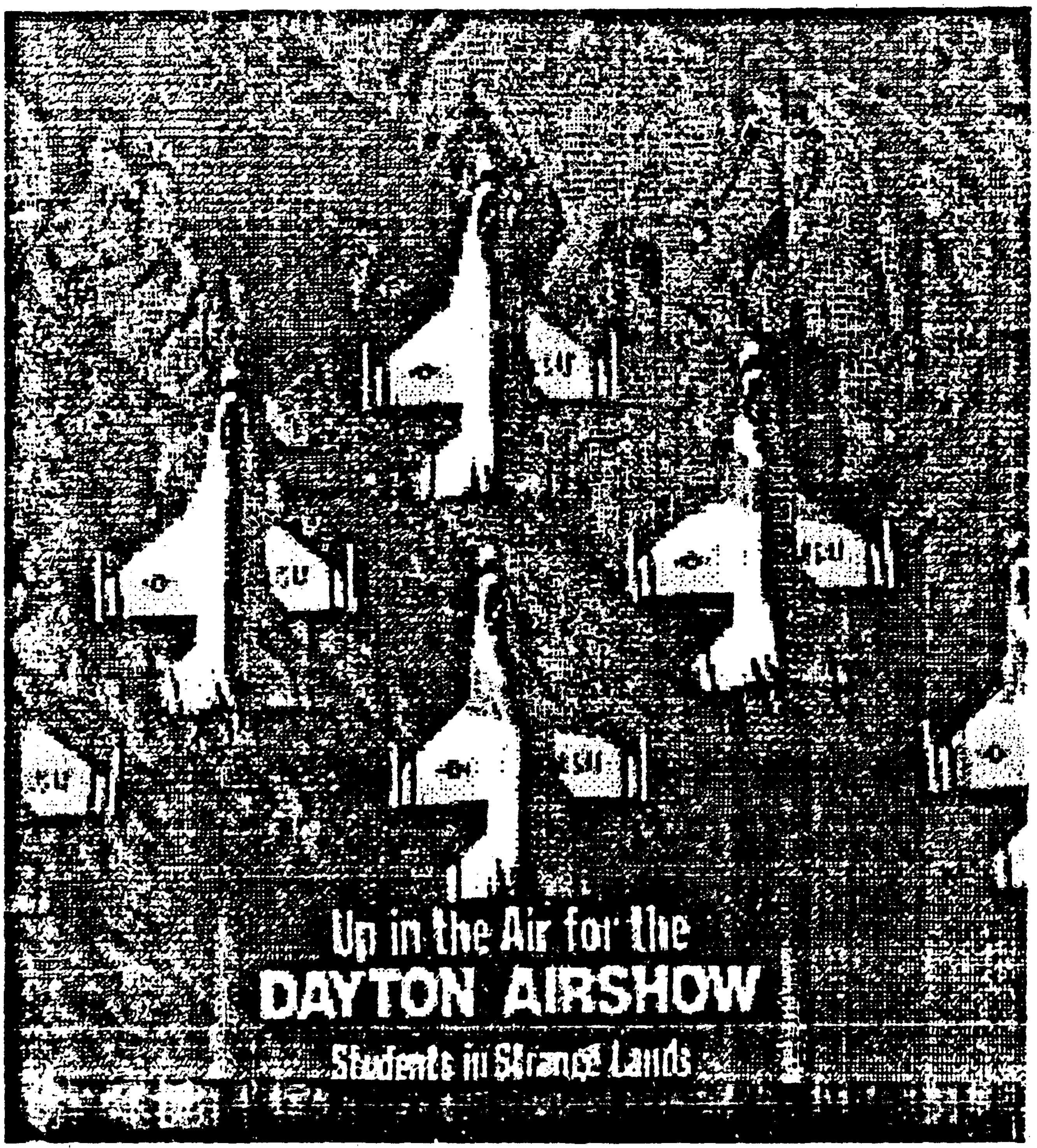


Figure 2.2 256 x 256 Pixel Representation of F-16's.

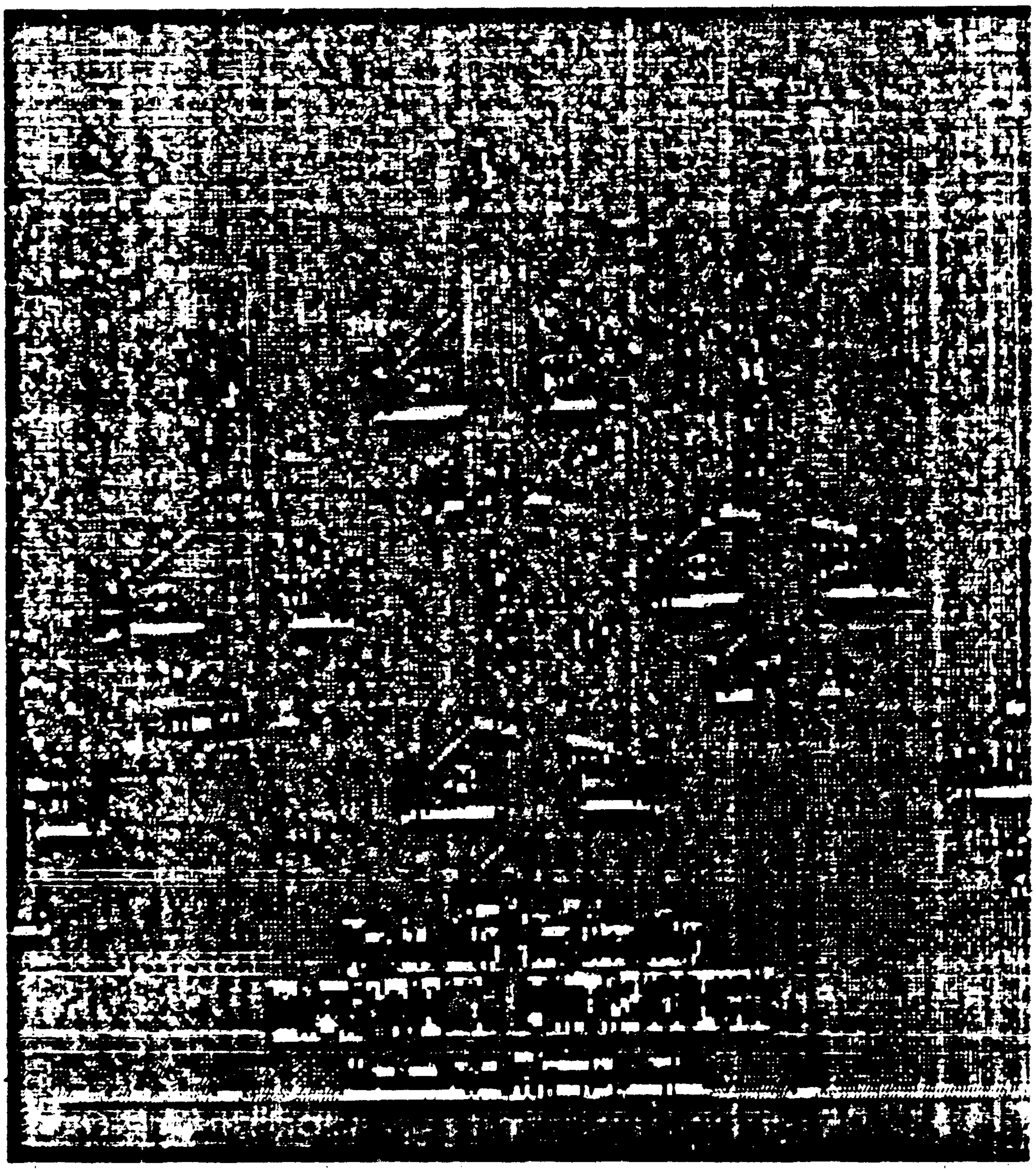


Figure 2.3 Edge Extraction in the Horizontal Direction

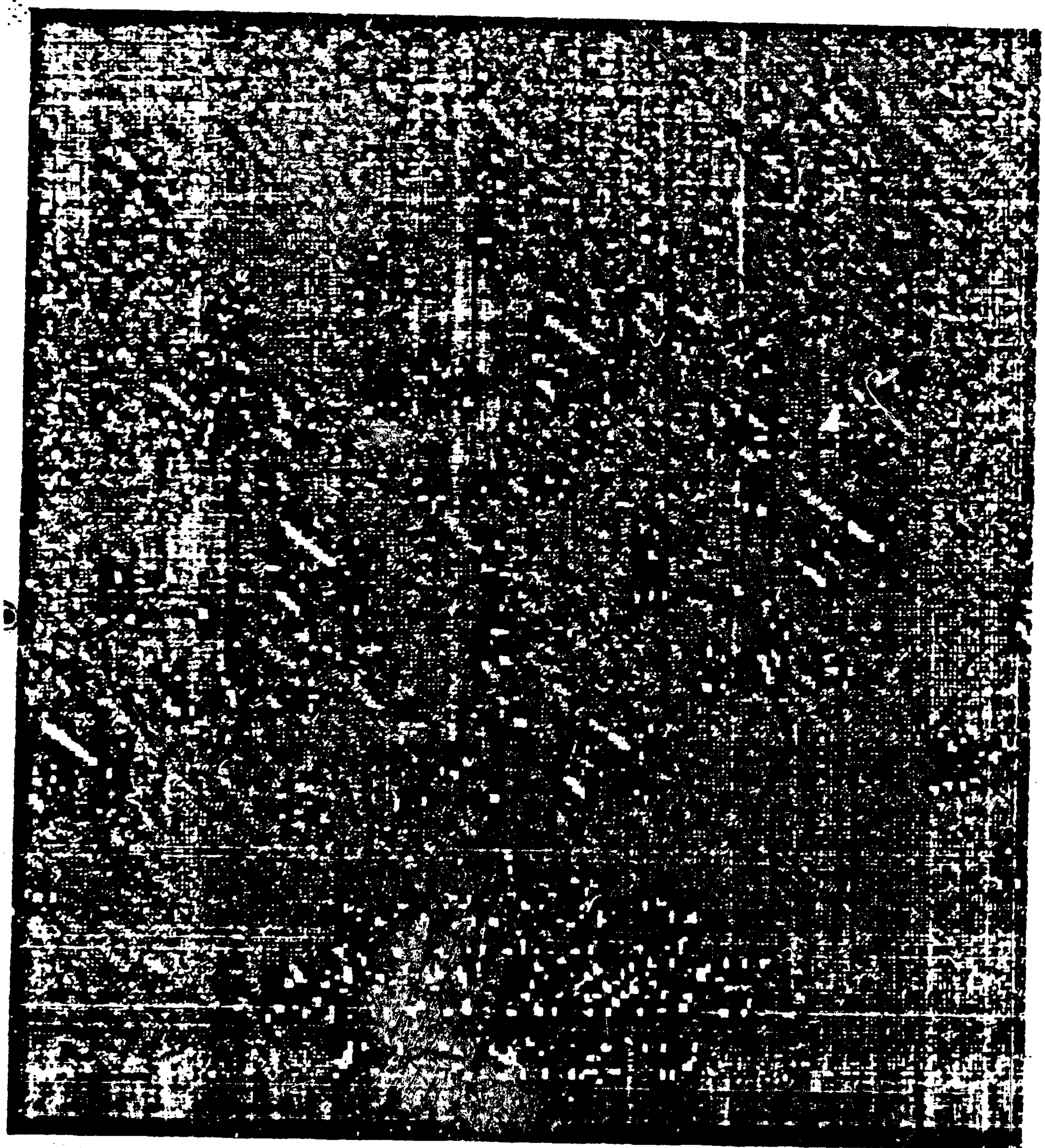


Figure 2.4 Edge Extraction in the First Diagonal Direction

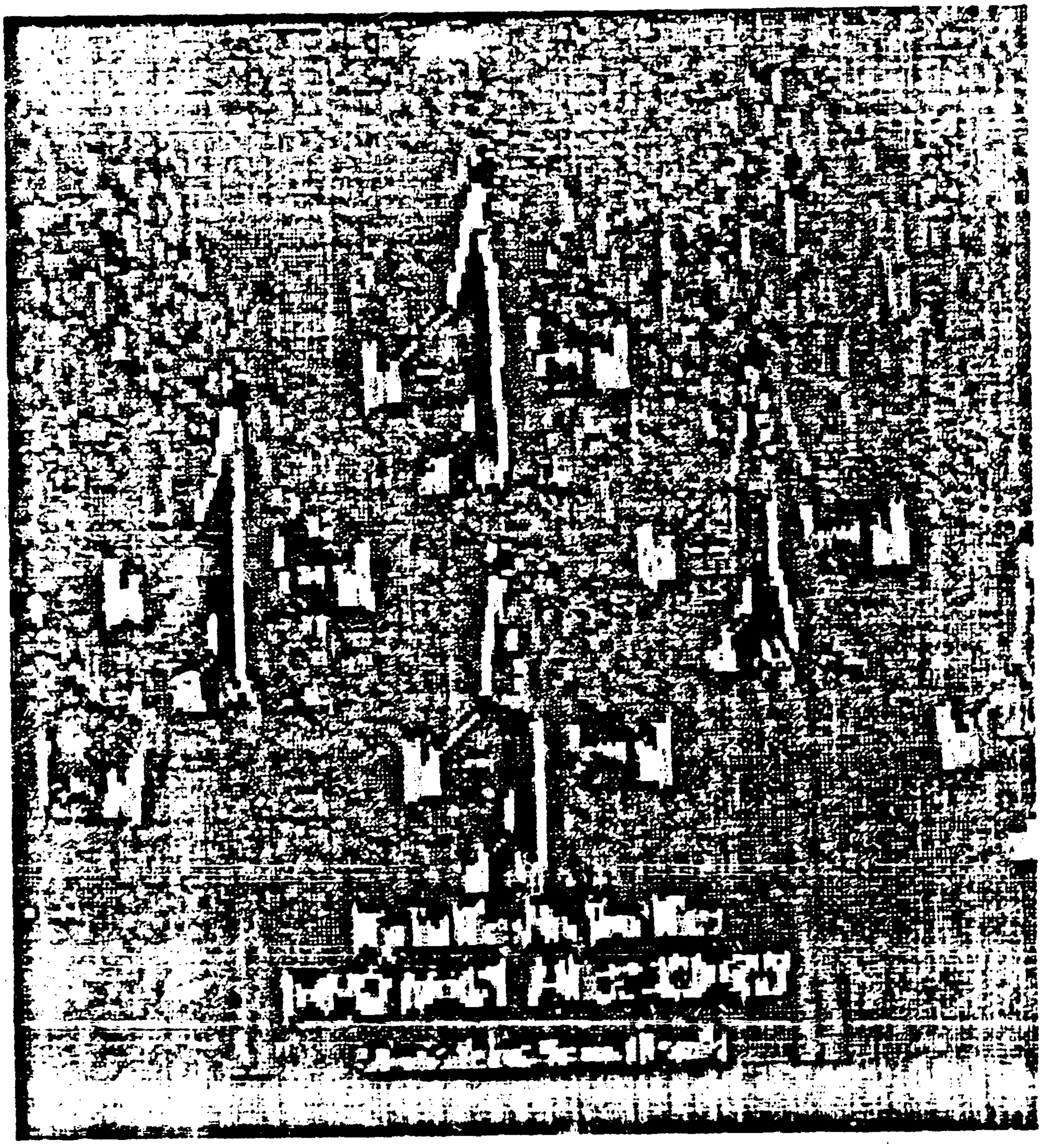


Figure 2.5 Edge Extraction in the Vertical Direction

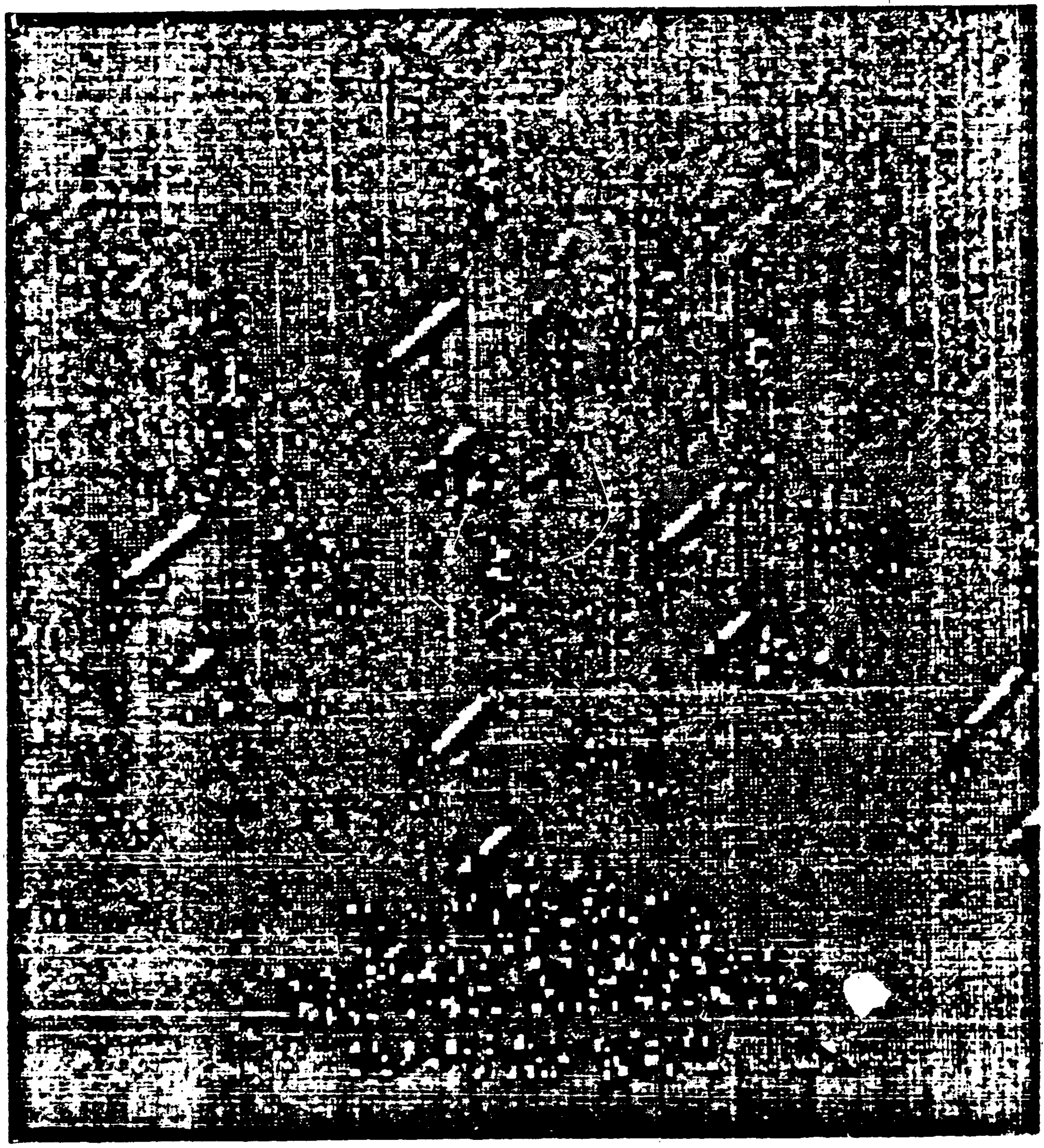


Figure 2.6 Edge Extraction in the Second Diagonal Direction

The scene, containing F-16s, in Figure 2.2 is the example used throughout the text of this thesis. This image was used for several reasons:

1. It contained multiple targets which met the previously stated constraints in the Assumptions.

2. It had a noisy background.

3. It contained high energy noise from the words artificially placed on the scene (the scene was originally a magazine cover).

The second step of the edging algorithm is the combining of the four output files into one output file which contains the edges in all directions. This is accomplished by the program COMB which combines the four output files of the program WEDGE. Figure 2.7 show the result of COMB performed on Figures 2.3-2.6.

The software for the Wedge preprocessing is contained in Appendix B.

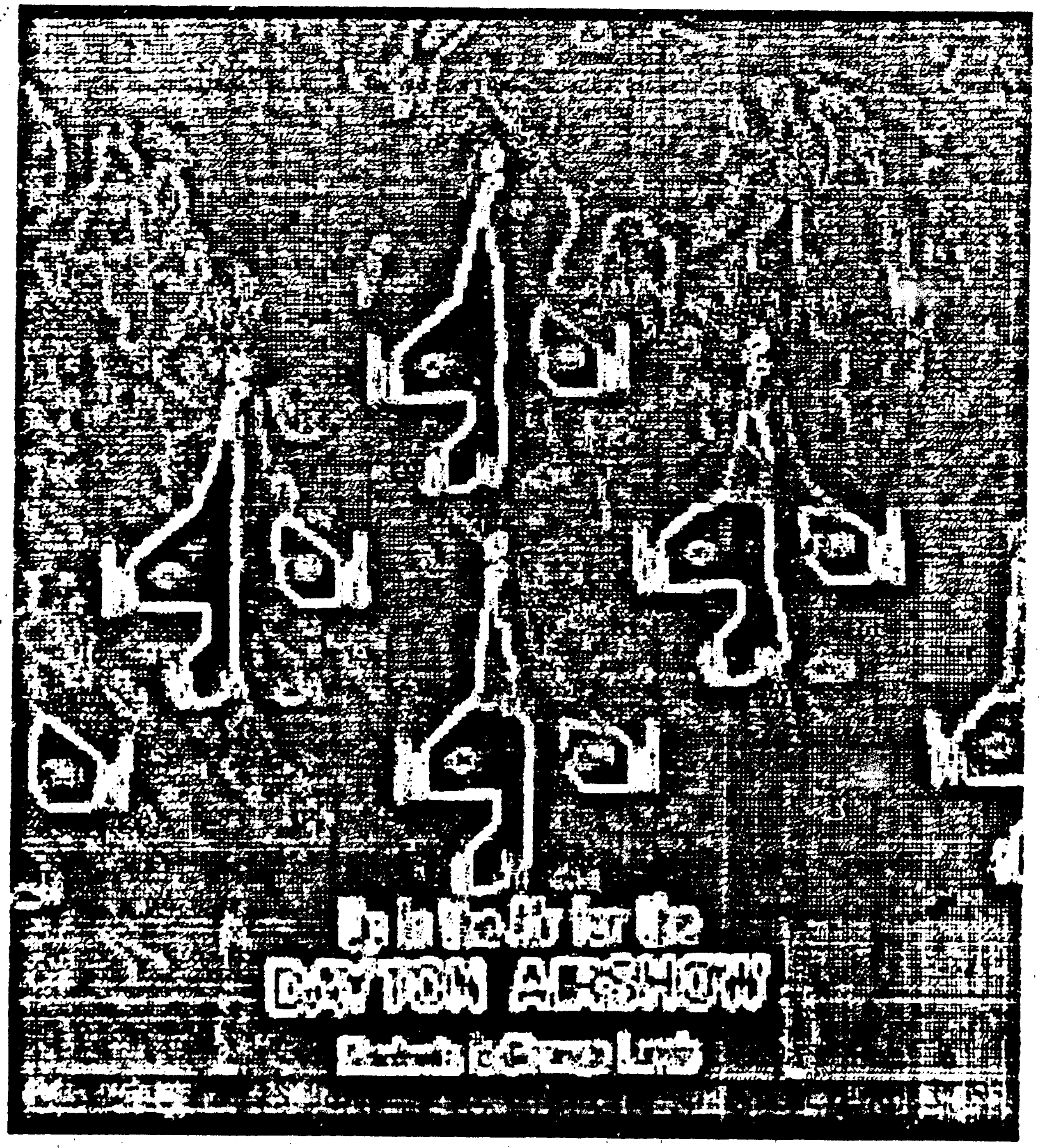


Figure 2.7 Complete Edge Extraction
(Combination of the Files of Figures 2.3-2.6).

III. FIRST PASS CORRELATION PROCESS

3.1 TEMPLATE GENERATION

Before the correlation process begins a template of the desired target must first be generated. The templates were first taken directly from the scene itself and later from other sources. In any case, the template will exist in some video file. From this video file some optional preprocessing using the Octek board may be used. The template must be in a rectangular area that is free of any unwanted noise. The program OCTEK has the ability to block out and replace any area of the video file with any other gray level (in this case 0 for black). Figure 3.1 shows the desired template information from Figure 2.7 after the noise was removed using OCTEK.

The template video file is produced by the program SPIN. This program creates a new video file which places the template at the origin of the coordinate system. In the frequency domain the template assumes to exist in a wrap-around-space. This means if an image moves off the screen at the top, it will reappear at the bottom; if it moves off the screen to the left, it will reappear on the right, et cetera.

Using this concept, SPIN places the original template at the origin of the new scene. This will result in the correlation peaks having the same coordinates of their corresponding targets or noise. This is done by quartering the original template and moving the four portions to the opposite corners in the new template scene. The area containing the template information is defined by the row-column coordinates of the upper-left and lower-right corners. (The coordinates are determined with an OCTEK function as shown in Figure 3.2.) SPIN transfers the corners of the quartered template image into the opposing corners of a black video file (see Figure 3.3). The final video file of the template is shown in Figure 3.4.

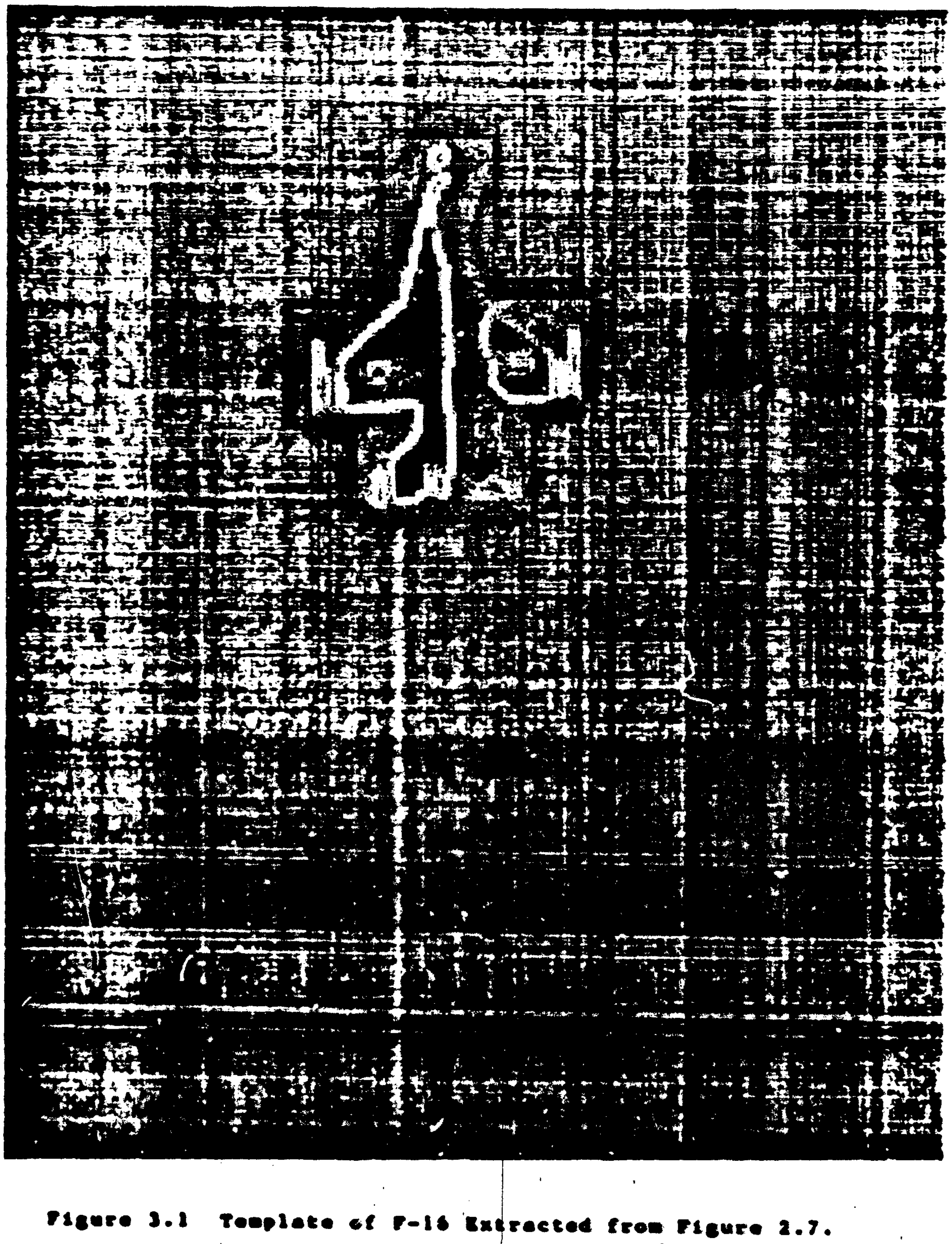


Figure 3.1 Template of F-16 Extracted from Figure 2.7.

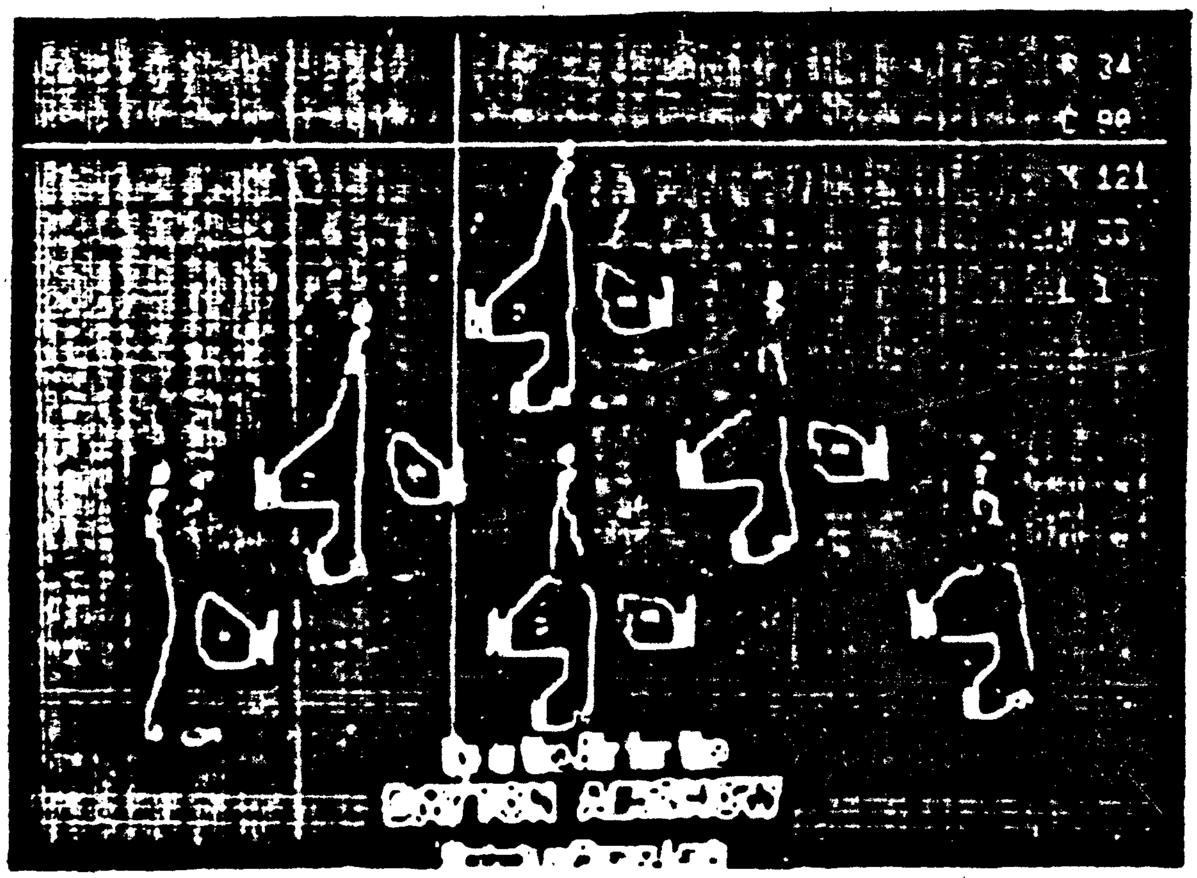


Figure 3.2 OCTEK Cursor Movement (Function #4).

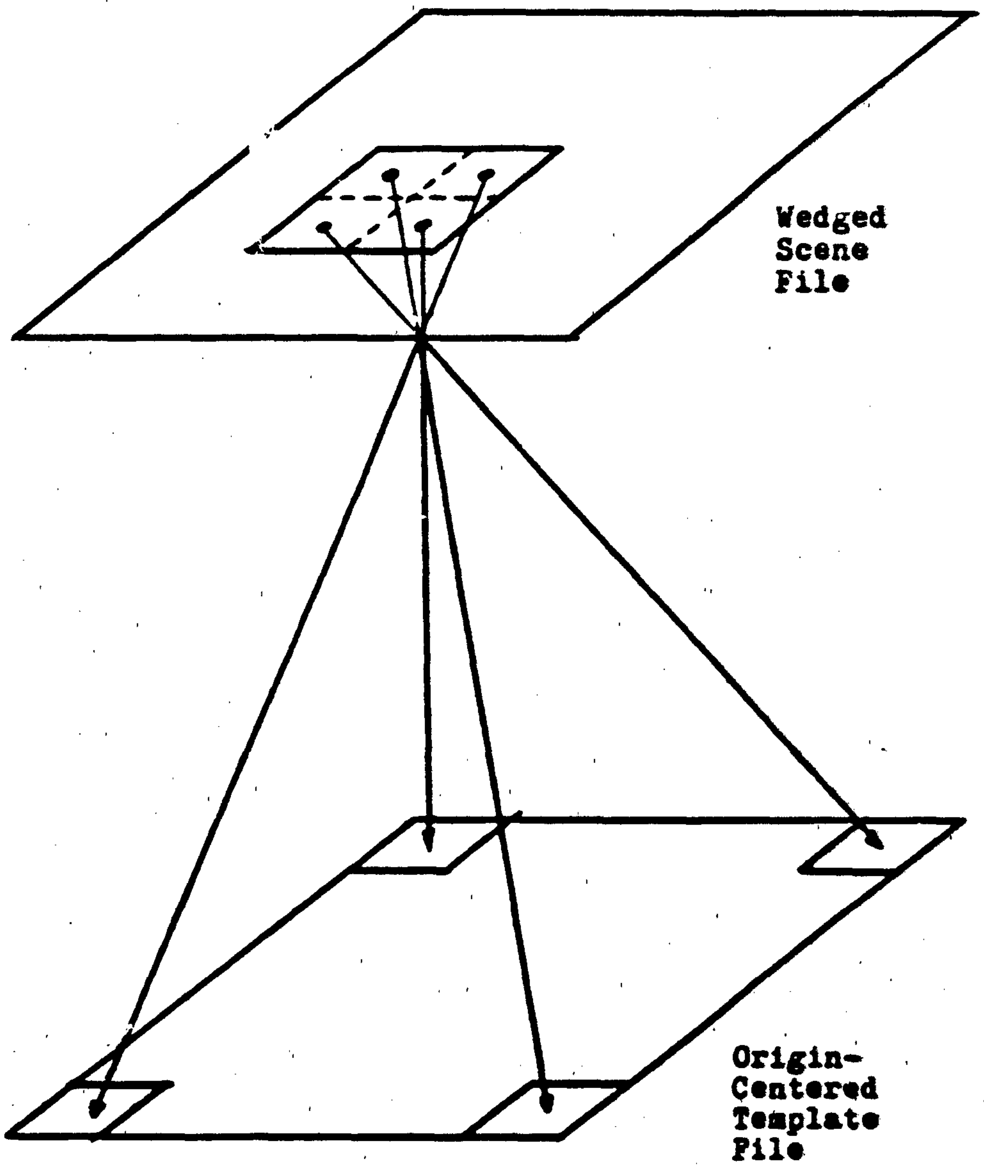


Figure 3.3 Image Information Movement due to SPIN.

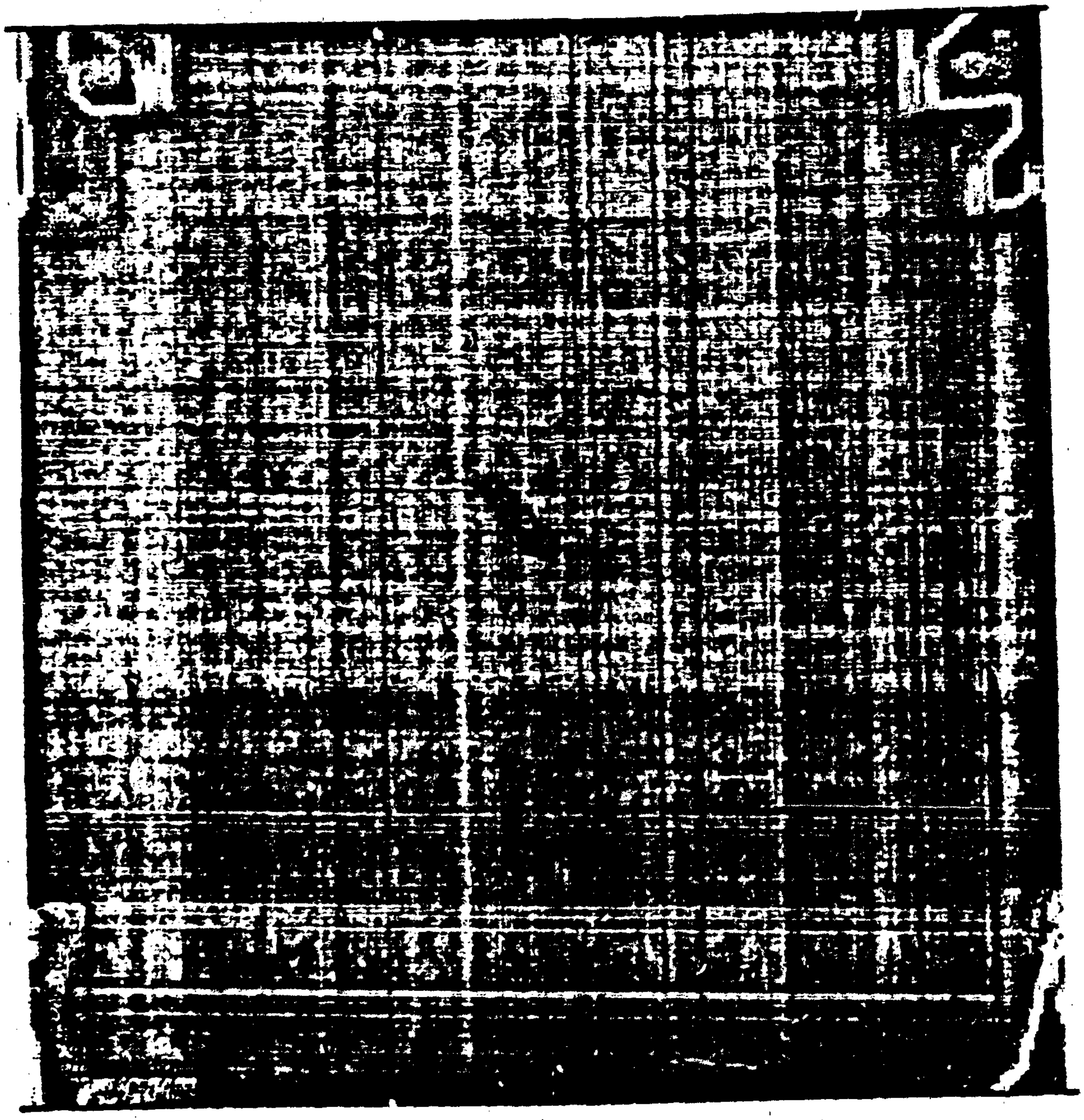


Figure 3.4 Final Video File of Template of F-16.

3.2 TWO DIMENSIONAL FAST FOURIER TRANSFORM

The correlation of the template with the scene takes place in the frequency domain rather that in the spatial domain. This is accomplished by the multiplication of Fourier transforms. A correlation in the spatial (or time) domain is equivalent to performing a conjugate multiplication in the frequency domain (Ref 6). This is shown in the following equation:

$$f_1(t) \star f_2(t) \iff F_1(f)^* F_2(f) \qquad (3\text{-}1)$$

where $f_1(t)$ and $f_2(t)$ are time domain functions, $F_1(f)$ and $F_2(f)$ are Fourier transforms of those functions and * is the complex conjugate operator.

Equation 3-1 may be written in the two-dimensional spatial domain, rather than in the time domain. This would yield equation 3-2:

$$f_1(i,j) \star f_2(i,j) \iff F_1(f)^* F_2(f) \qquad (3\text{-}2)$$

where now $f_1(i,j)$ and $f_2(i,j)$ are spatial functions with the coordinates (i,j). To do this, the video file must first be in complex form. The program VDTOCP (Video-To-Complex) does this by creating a new file which contains the same data of the video file but in complex form with the imaginary parts set equal to zero. In this new format the Two Dimensional Fast Fourier Transform (2DFFT) may be performed.

The 2DFFT is performed by the program DIRECT. It is this procedure which requires the much computation time. The transform of a 256x256 complex file takes approximately 11 minutes on the Eclipse background (at least 22 minutes on the foreground, depending on the job load). The procedure of VDTOCP and DIRECT is performed on both the file containing the origin-centered template and the file containing the scene to be searched.

3.3 FILE MULTIPLICATION

To correlate the two 2DFFTs, a point-by-point conjugate multiplication is performed. This is done with the program VCM (Variable-Conjugate-Multiplication). The result is in complex form.

An alternate program was later used to perform the file multiplication. This was one which also filtered the transform of the template as it multiplied it with the transform of the scene. This is discussed later in this chapter under Section 3.6.

3.4 INVERSE TWO DIMENSIONAL FAST FOURIER TRANSFORM

The output file of VCM contains the multiplication of the 2DFFT of the template file and of the scene file. The next step is to perform an inverse 2DFFT. The program INVERSE does this computation. This function also takes about 11 mintues on the Eclipse background. The output of INVERSE is still in complex form.

The resultant output file is now converted to a packed video form with the program CPTOVD (Complex-To-Video). This program scales the complex data and converts it into integer values from 0 to 15 so that it may be displayed by OCTEK.

3.5 OUTPUT

The video file containing the correlation results may now be displayed on the viewing screen with OCTEK (or by hard copy with the program PIX). Figure 3.5 shows such an output. Sharp white areas indicate possible correlations with targets. The peaks may also be false correlation with high energy noise in the scene.

Figure 3.6 is a photograph of a display monitor showing the same correlation output (after the grey levels were reversed). This is included in order to compensate for the lost resolution of the PIX output after reduction.

Due to characteristics of this correlation in the frequency domain the peaks will occur at the location of the template origin durring the line-up between the template and the target. In order that the peaks correspond with the actual location of the targets the template and the origin must exist at the same location. It is for this reason that the template information was centered at the origin of the template video file.

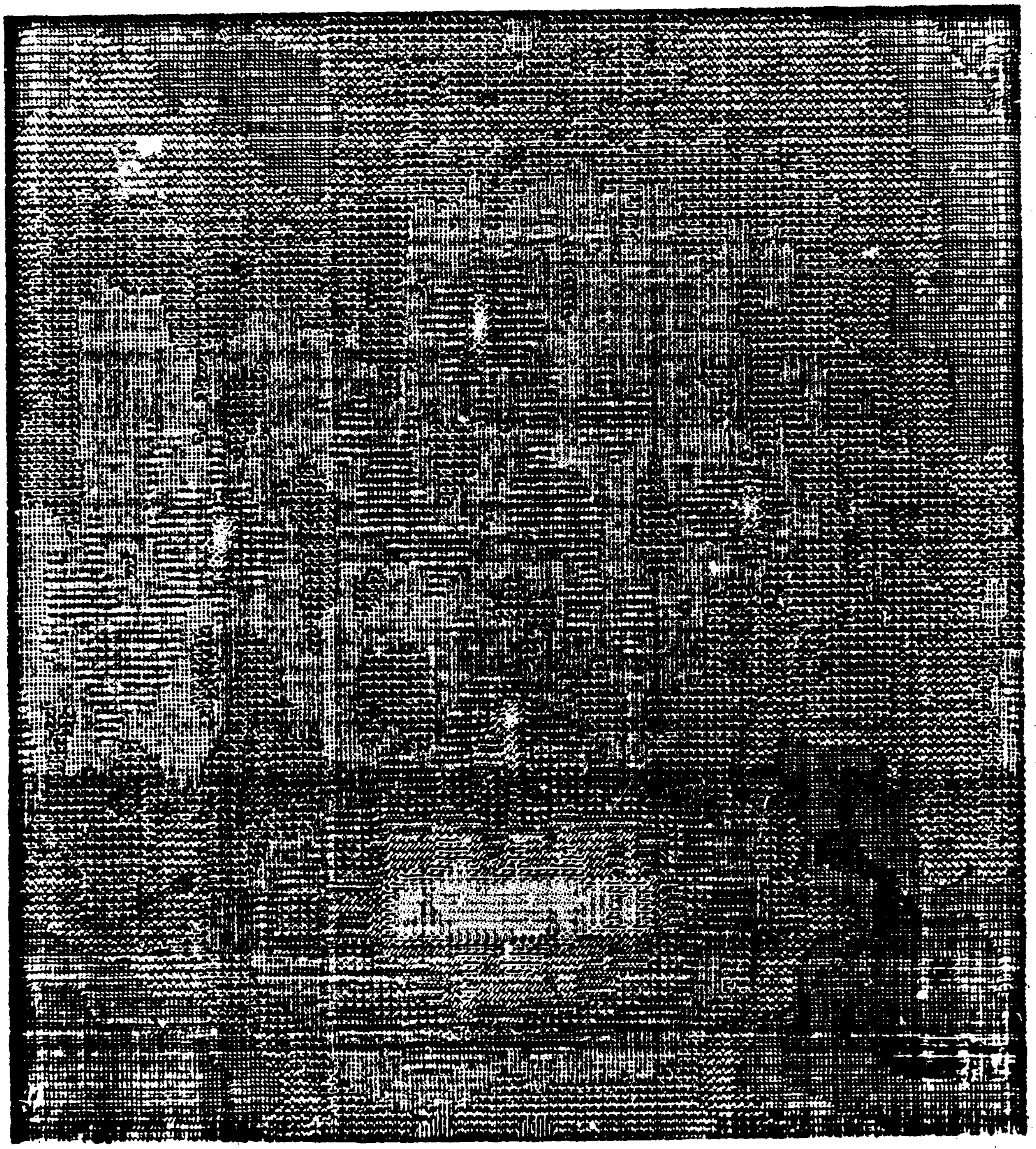


Figure 3.5 Correlation Output of Target Scene (Figure 2.7) with Origin-centered Template of F-16 (Figure 3.4).

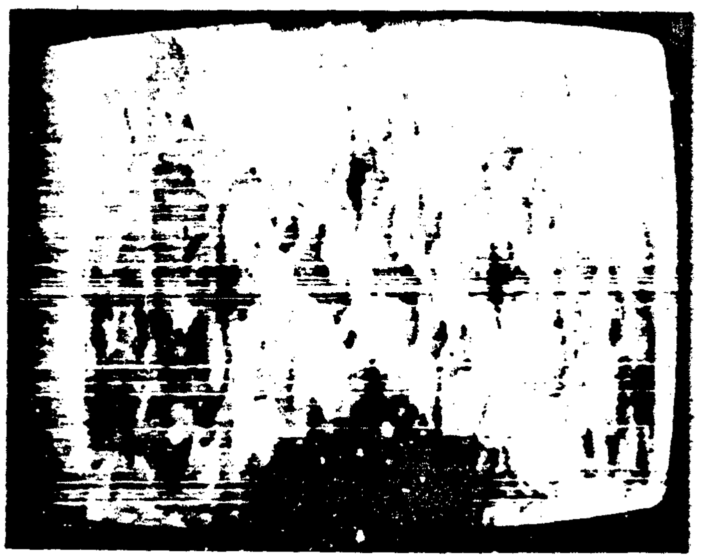


Figure 3.6 Display Monitor Output of Figure 3.5.

3.6 FOURIER FILTERING

As mentioned earlier in this chapter the templates were originally taken from the multiple target images being examined. However, in an actual operating system this would not be practical. The entire process was redone on this test scene, but a more general F-16 template was used.

An image of an F-16 with the same perspective was digitized using OCTEK. This new template differed slightly in size and orientation compared to the targets in the scene. The most drastic difference was the shading. The airplanes in the original target scene were shaded from the side (Figure 2.2). This produced a sharp edge along the fuselage of the aircraft (Figure 2.7) after the Wedge edge detector was applied. The new generalized template was evenly shaded so that the Wedge operator produced a normal outline of the F-16. (see Figure 3.7b). As a result, the new template had a different shape than the targets (shown in Figure 3.7a).

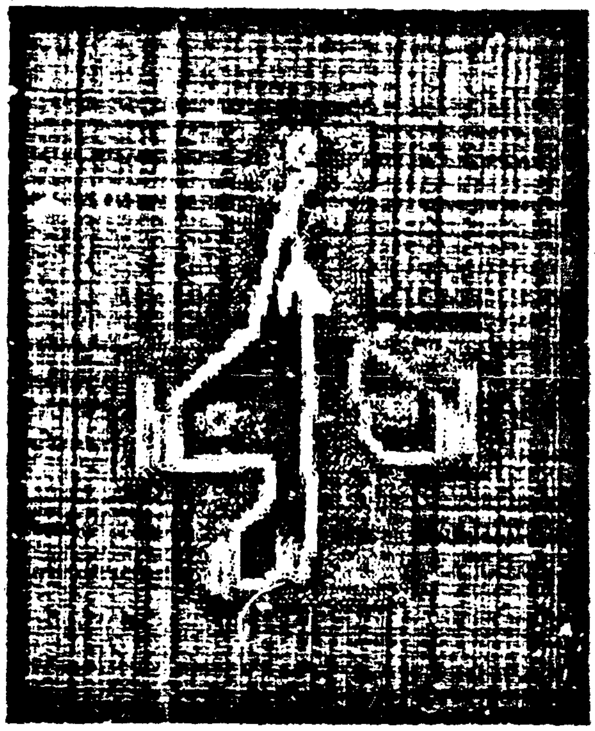
In order to match a template with non-identical targets a filtering technique was used. Without the filter, a non-optimum output was produced. Figure 3.8 is the output of the template of Figure 3.7b correlated with the Wedged scene of Figure 2.7. A photograph of the display monitor output is again included ,in Figure 3.9, to show clarity of the grey levels.

In theory, the filter is to be applied to the template transform and then the multiplication of template with scene can take place. However, this would result in an additional one-half megabyte file being produced for each filtered version of the template. This filtered version is refered to as T' in equations 3-3a and 3-3b. These equations show the theoretical application of the filter:

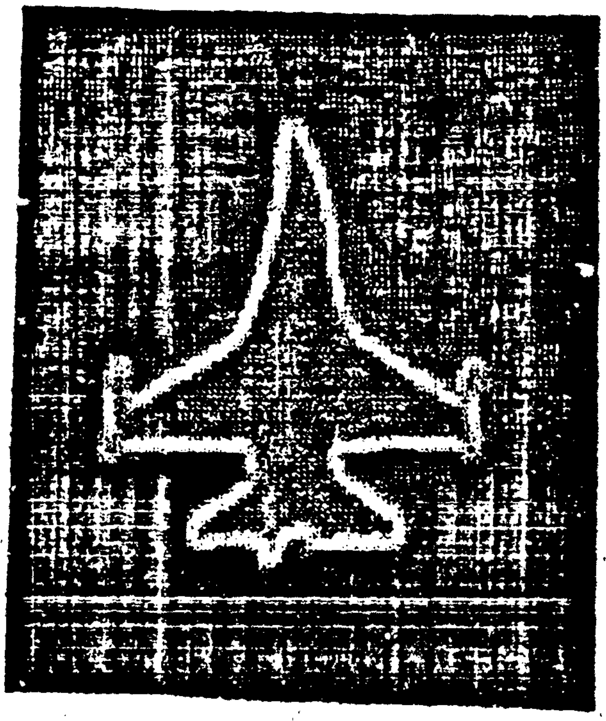
$$T' = T^*F \qquad (3\text{-}3a)$$

$$O = S^*T' \qquad (3\text{-}3b)$$

where T is the template file, F is the filter, T' is the filtered template file, S is the scene file and O is the output file. (The operator, *, represents complex conjugate multiplication.)



a



b

Figure 3.7 Wedged F-16 Templates.
a) Template from the Scene
b) Normal, Generalized F-16 Template.

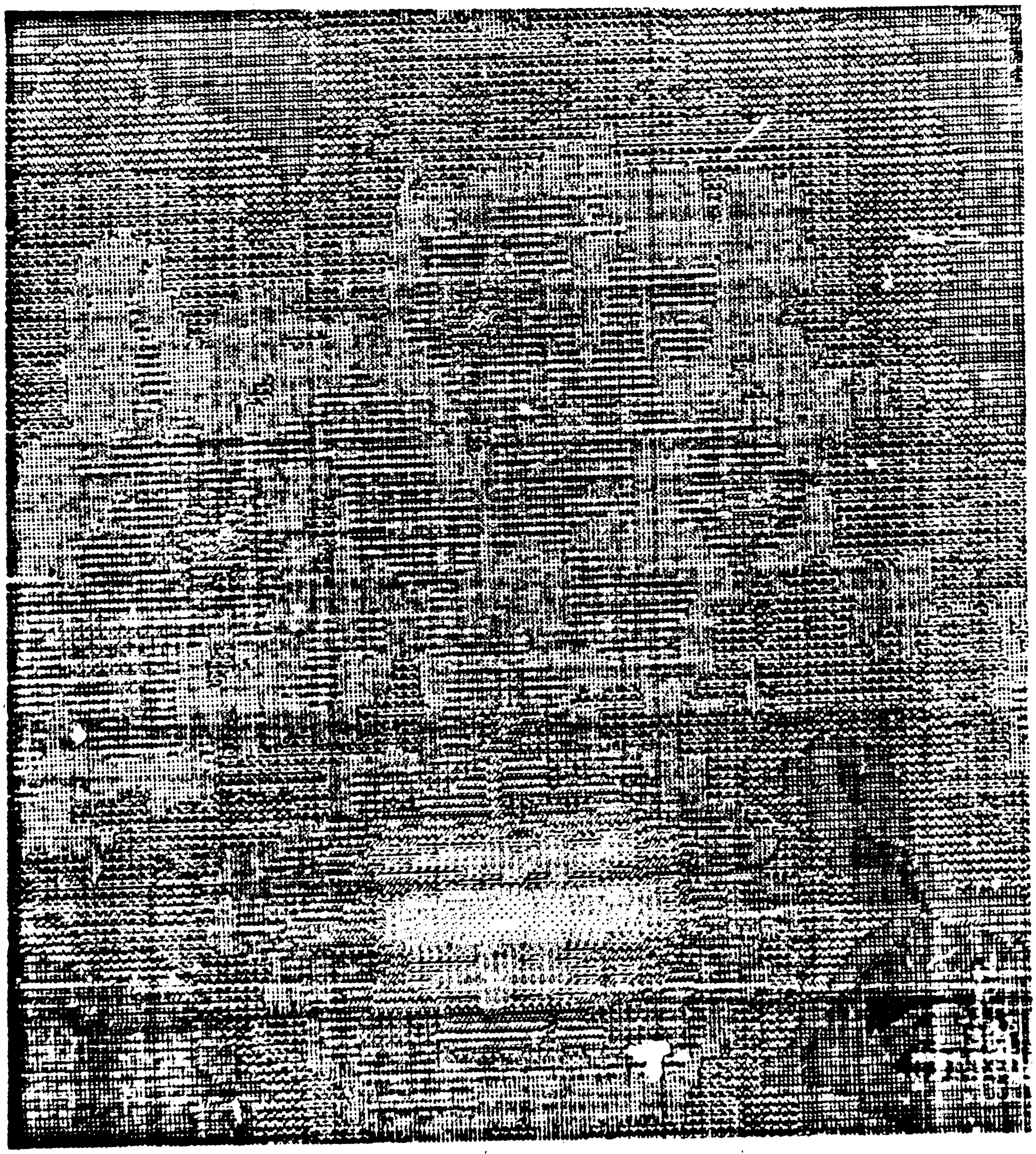


Figure 3.8 Correlation Output of Target Scene (Figure 2.7) with General F-16 Template (Figure 3.7b).

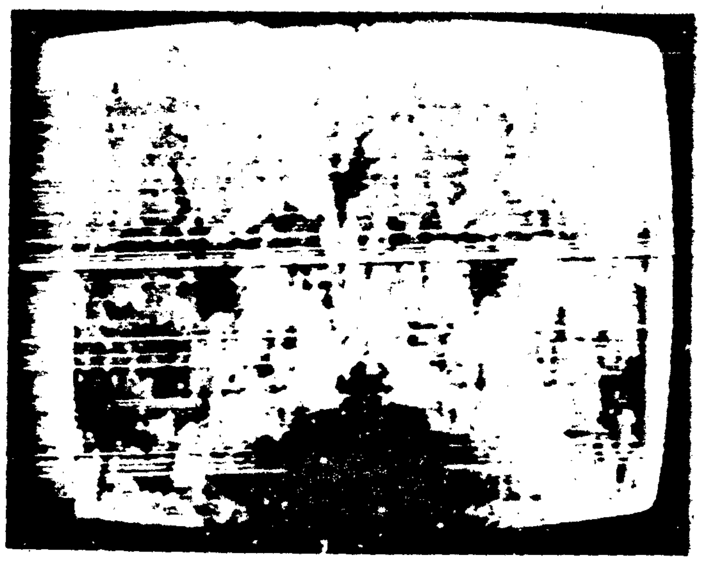


Figure 3.9 Display Monitor Output of Figure 3.8.

Due to the properties of complex congugate multiplication, as well as all multiplication, substitution can be used. The result of this substitution is given in equation 3-4:

$$O = S^{*}T^{*}F \qquad (3\text{-}4)$$

This operation is preformed with the program FVCM (Filtered-Variable-Conjugate-Multiplication). The program FVCM does the same conjugate multiplication as does the program VCM (Section 3.3) but with the application of a variable sized filter.

The filter used was one which only multiplied the dc term and a specified number of lower harmonics. This is done because it is the lower frequency harmonics which contain the main information of the scene, while the higher harmonics contain only minor details (Ref 4). Figure 3.10 shows the output with the application of a filter of pixel size 7x7 (dc term and first three harmonics). It should be noted that this was over filtered to the point where only the high energy noise produced correlation peaks. Figure 3.11 is the output with a more optimum filter, pixel size 39x39 (dc term and first 19 harmonics). The display monitor output is shown in Figure 3.12. This filter size was judged to be optimum for this type of scene. FVCM can operate with any user specified filter size from 7x7 pixels up to 255x255 pixels, in increments of eight pixels (7x7, 15x15, 23x23, 31x31, etc.). The increment of eight was needed due to the software techniques used.

After filtering, the standard first pass techniques were used. The output, such as Figure 3.11, was used to determine the locations of potential targets for the second pass analysis.

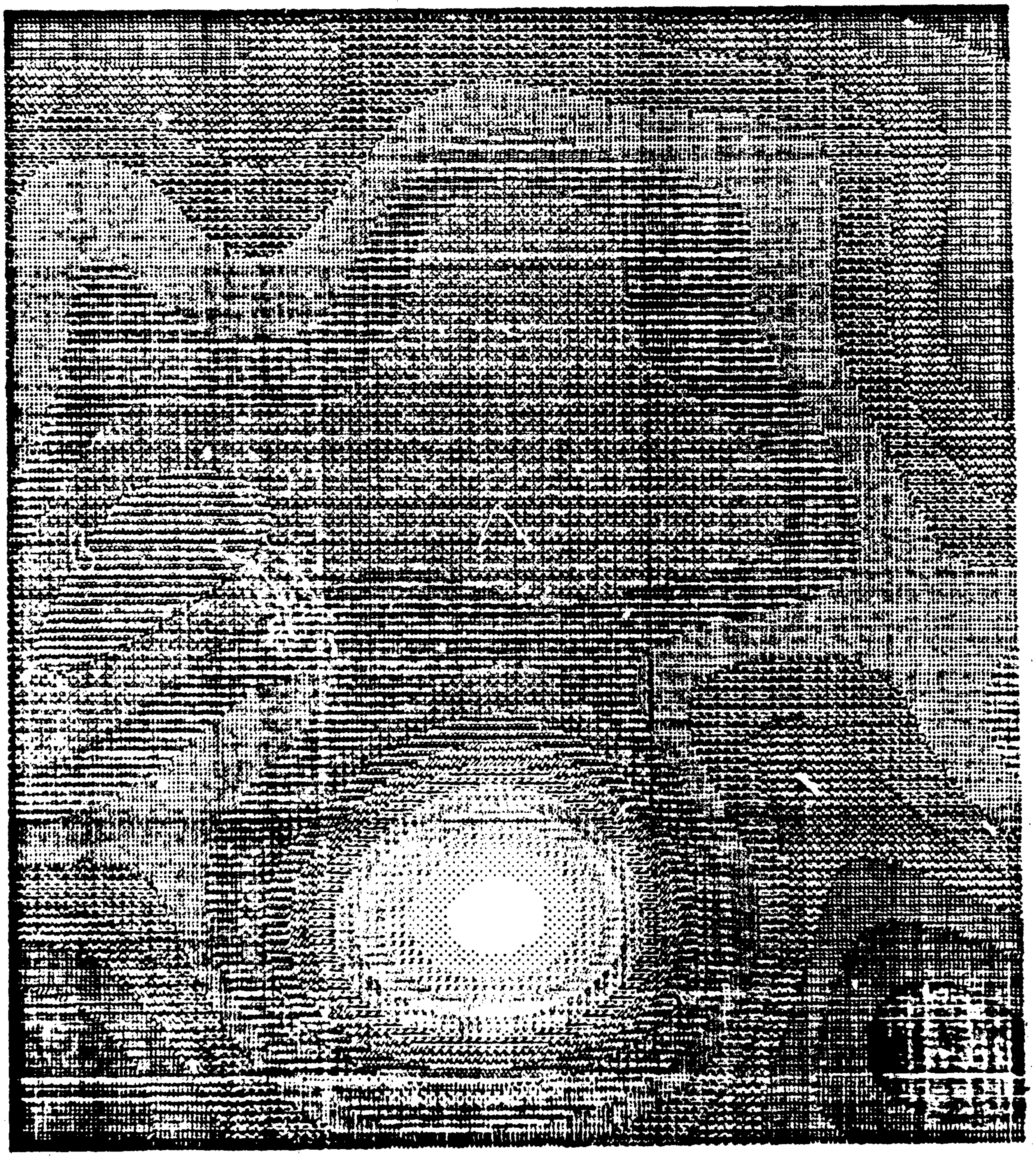


Figure 3.10 Correlation Output of Target Scene (Figure 2.7) with General F-16 Template (Figure 3.7b) using a 7x7 Pixel Filter (dc and first 3 harmonics). (Over Filtered)

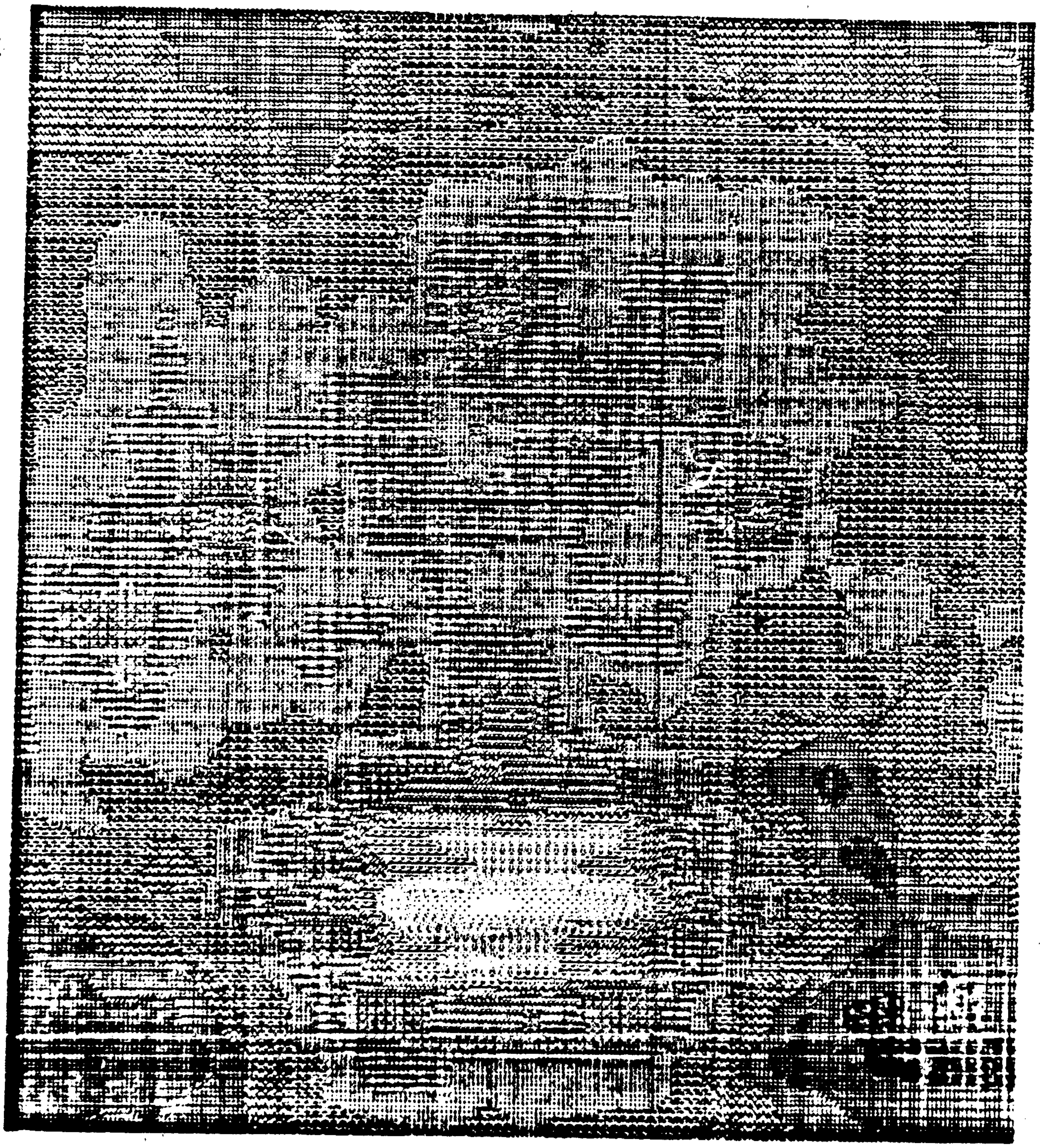


Figure 3.11 Correlation Output of Target Scene (Figure 2.7) with General F-16 Template (Figure 3.7b) using a 39x39 Pixel Filter (dc and first 19 harmonics).

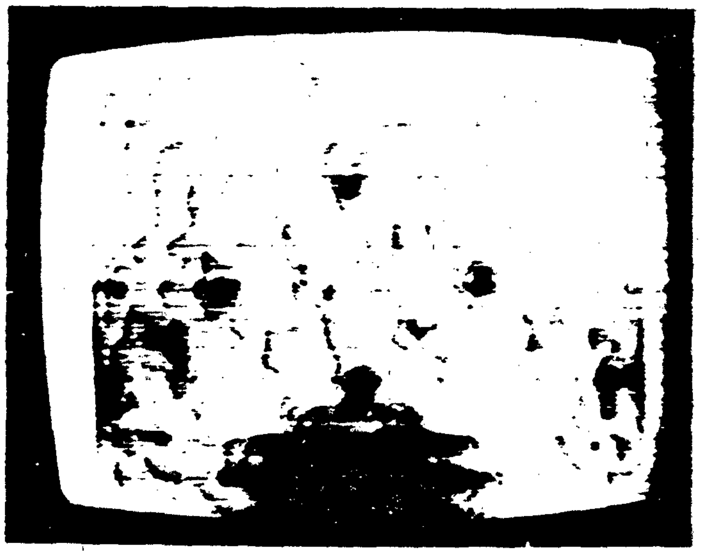


Figure 3.12 Display Monitor Output of Figure 3.11.

IV. SECOND PASS CORRELATION PROCESS

The basic process described in Chapter Three is repeated a second time to distinguish the correlation peaks as being targets or noise. Due to the characteristics of the correlation process, high energy noise will create a peak. The second pass is used to re-examine the potential target at the location of the correlation peaks. The unique feature of the second pass is that the video file containing this potential target is energy normalized to that of the template file.

4.1 FILE REGENERATION

For reasons of efficiency, portions of the Wedged scene were extracted from the 256x256 video file and placed into smaller video files. This was done for two reasons. First, certain areas in the scene could be close to more than one peak. This would mean that this area could be operated on by the normalization process more than once and therefore produce artifical errors.

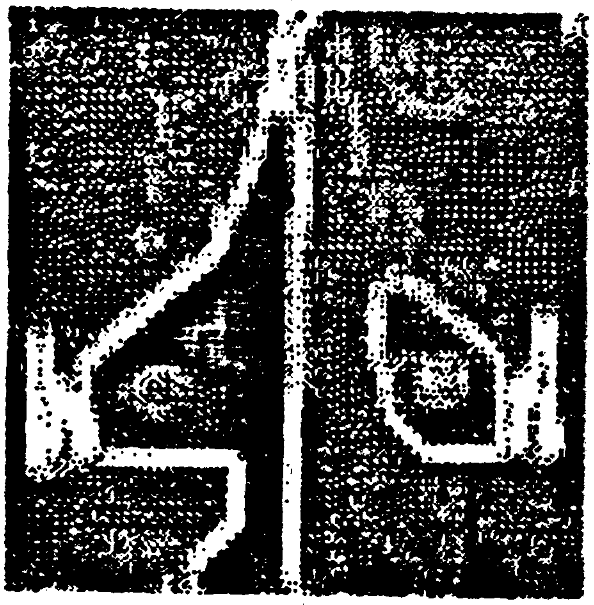
The second reason was the time factor involved. Since each file could contain only one area which was energy normalized, a file which contained multiple peak would have to be normalized and recorrelated for every peak. The number of peaks for potential targets could typically be as high as ten. Considering the fact that the total process took about 40 minutes for computation of 256x256 files, this size was prohibitive. When smaller files were used, the computation time for the programs DIRECT and INVERSE was considerablly shorter.

The small files were created with the program SMALL. This program reproduced the video information from one file into the smaller packed video file. The source file was the Wedged image of the scene. The location of the potential target was found by using OCTEK to determine the row-column coordinates of the corresponding correlation peak. The size of the smaller video files was limited to 128x128, 64x64, and 32x32. This was due to the constraints on the FFT programs DIRECT and INVERSE for which there was no availible source code. (Attempts were made to recreate such programs, but the disk I/O time on the Eclipse was too lengthly.)

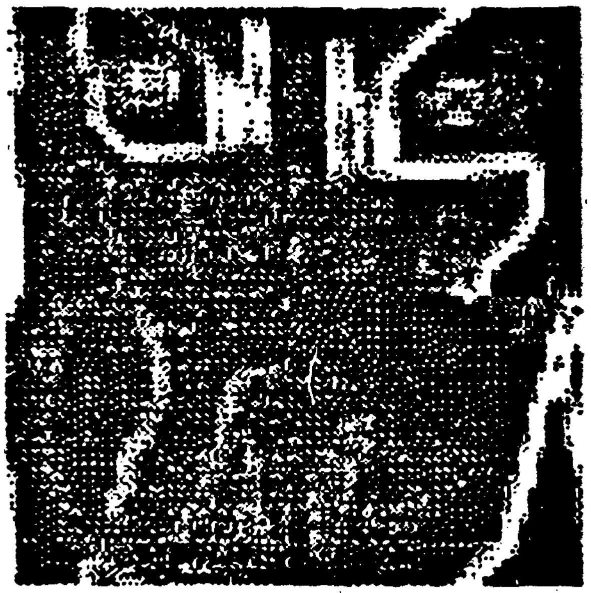
The typical size of a smaller file was 64x64. This enabled the programs DIRECT and INVERSE to operate in 22 seconds. With this turn around time, re-evaluations of numerous potential target peaks could be made. Smaller video files of all potential targets were made, as well as a smaller video file of the template. It was the size of the template image which governed the size of all these smaller files.

During this second pass, the template must again be centered at the origin of the file. This was accomplished by the program VSPIN (Variable-Spin), which this time did not require the black background as did SPIN, since there was no file space not covered by template image information.

Figure 4.1 shows the smaller template before and after it was centered at the origin with VSPIN. Figure 4.2 shows the files corresponding to the correlation peaks (true and false) of Figure 3.5. These smaller files were displayed with the program VID, saved with OCTEK, and printed with PIX.



a



b

Figure 4.1 Small Template File. a) Before VSPIN and b) After VSPIN (No reduction of PIX output).

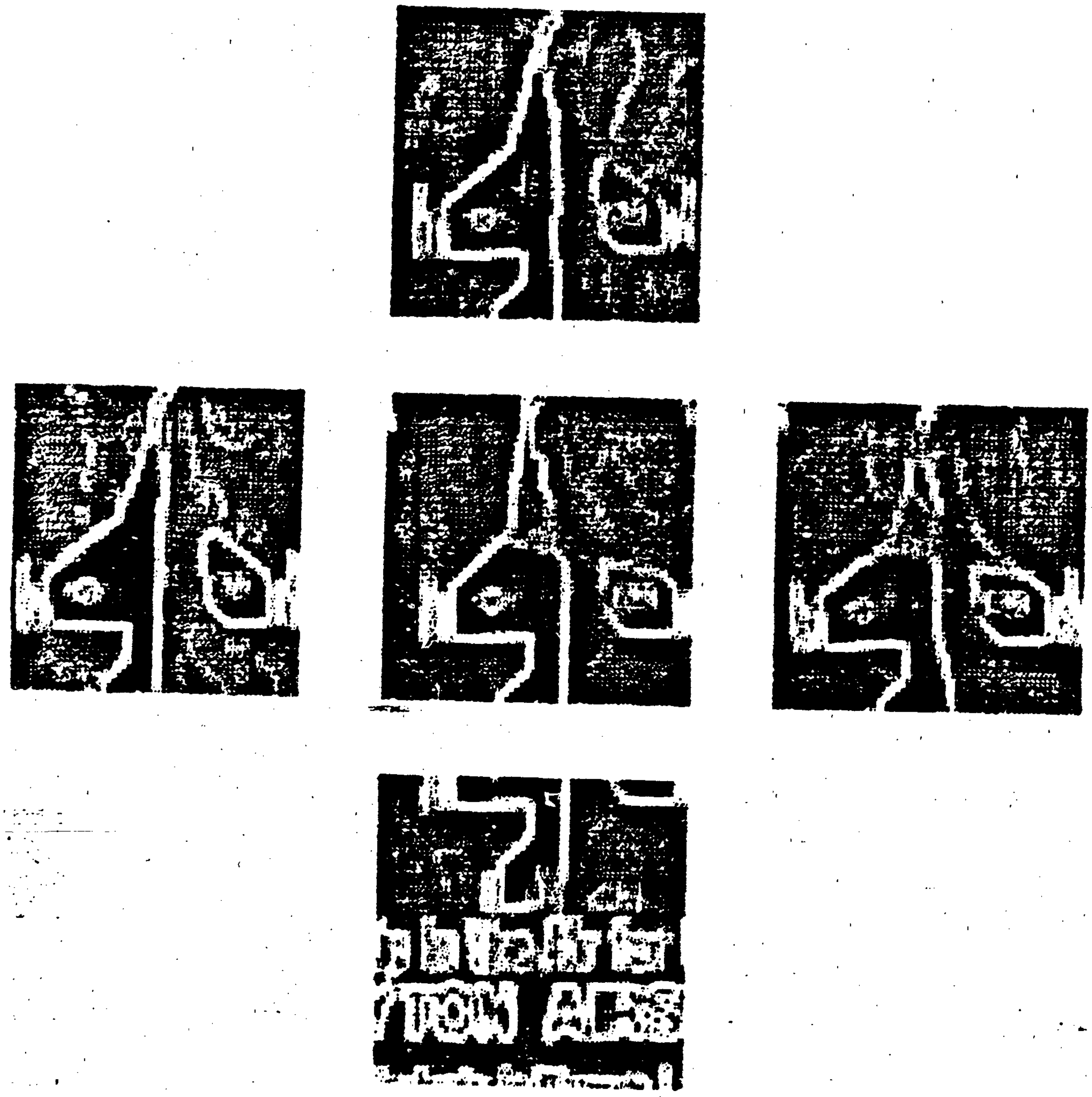


Figure 4.2 Small Files Corresponding to Peaks

4.2 ENERGY NORMALIZATION

The key to this second pass is the normalization of the energy in the video files of the potential targets. After energy normalization, recorrelation should still produce a peak for a true target but produce no peak for high energy noise. The energy in the video images is defined as being the summation of the squares of the pixel values (intensity). This is shown mathematically in equation 4-1:

$$\text{ENERGY} = \sum_i \sum_j I^2(i,j) \qquad (4\text{-}1)$$

where I is the intensity of the pixel at the point (i,j).

The normalization routine used was one which set the energy of the target file equal to that of the template file. Since only integer files were used, the energies could not be set to unity. Therefore, a pseudo-unity was used which was equal to the energy of the template file. The software to implement this routine is contained in the program NORM.

This program creates a third file which contains target information that may be only partially normalized. This original target file is multiplied by a factor that is the square root of the ratio of the template energy to the target energy. This is given in equation 4-2:

$$\text{factor} = \sqrt{\frac{Ep}{Eg}} \qquad (4\text{-}2)$$

where Ep is the energy of the template file and Eg is the energy of the target file.

Each new value is then converted to an integer. Due to round-off error in the integer conversion, the process is automatically repeated until no futher accuracy can be achieved.

NORM will not only decrease high energy noise, thus eliminating a false peak, but it will also enhance the energy of a low energy, target and increase the correlation peak. Figure 4.3 contains some of the video files of figure 4.2 after energy normalization. Note that the first two files show litte change, while the third shows a significant reduction in intensity due to the abundance of high energy noise. Later, in Figure 4.4, it will be shown that recorrelation with a true target does in fact sharpen the peak while noise produces a less defined peak.

4.3 RECORRELATION

The next steps of the second pass correlation process are very similar to those of the first pass. This time everything is redone but with two differences: First, smaller files are used (typically 64x64), and second, a complete recorrelation is done for each potential target area.

The smaller video files are declared complex by the program VTC (Video-To-Complex) which operates on variable sized files. This is done to the template file as well as the target files. The 2DFFT is again performed by DIRECT.

Each of the resulting complex target files are multiplied with the complex template file using the program VCM (see Section 3.3). The filtering multiplier, FVCM, has an option to filter 64x64 complex files with a first seven harmonic filter, but it was not used for reasons to be explained in the Results section (Section 5.4).

The output of VCM under goes Inverse Two Dimensional Fourier Transformation with the program INVERSE. The result is converted back to packed video with the program CTV (Complex-To-Video) which also operates on variable sized files.

The final output of the recorrelation can be displayed on the video screen with the program VID, saved with OCTEK and printed with PIX. Figure 4.4 shows such an output. Due to the nature of the program CTV, there will always be some type of peak but its location and uniqueness reveals the level of correlation (see Chapter Five). Visual inspection of these video files indicate whether or not a true target was present or if the first pass correlation peak was created by high energy noise.

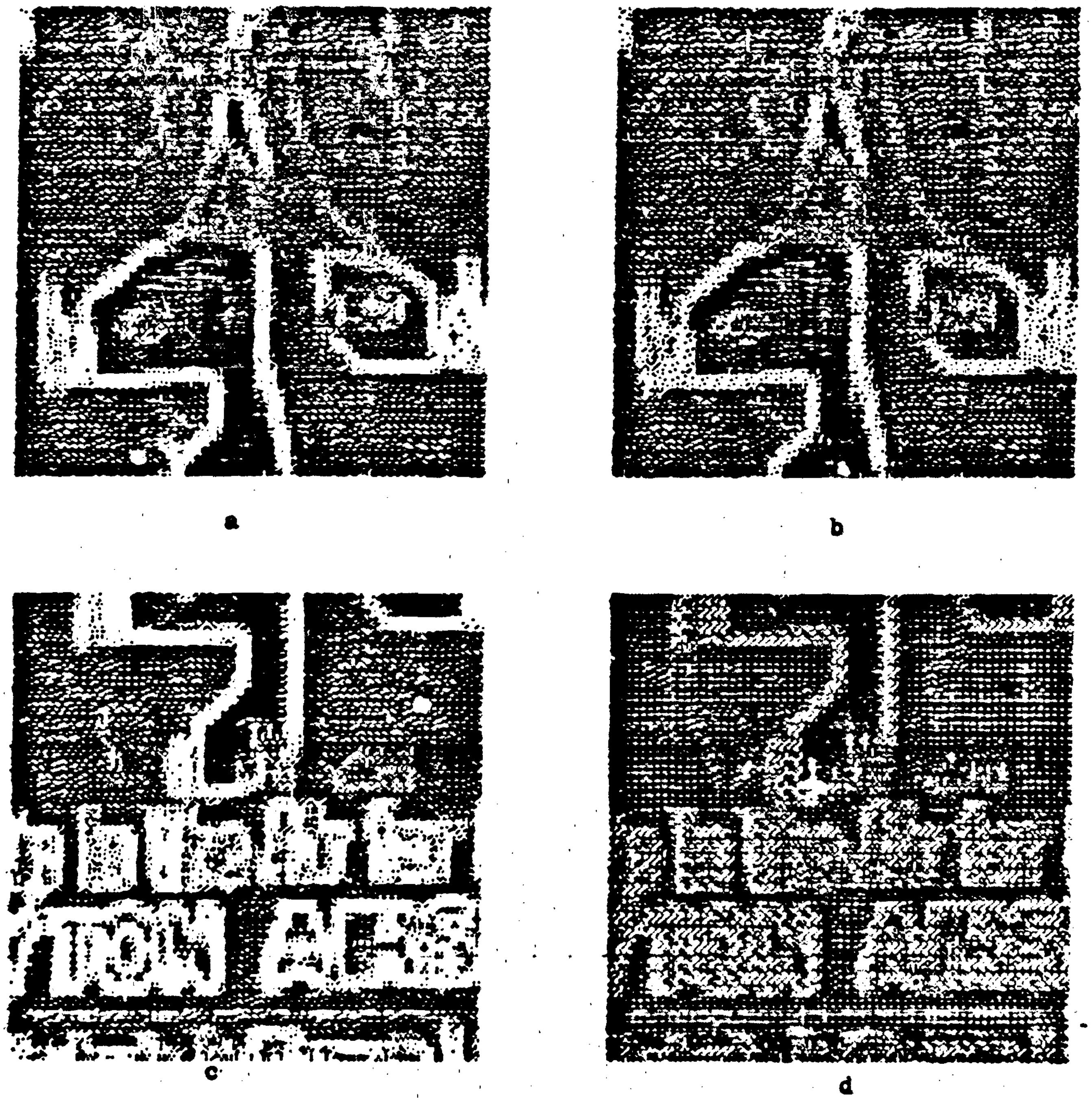
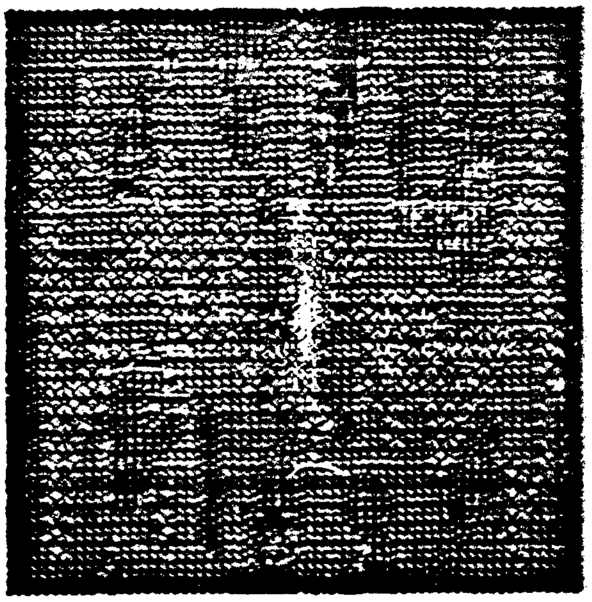
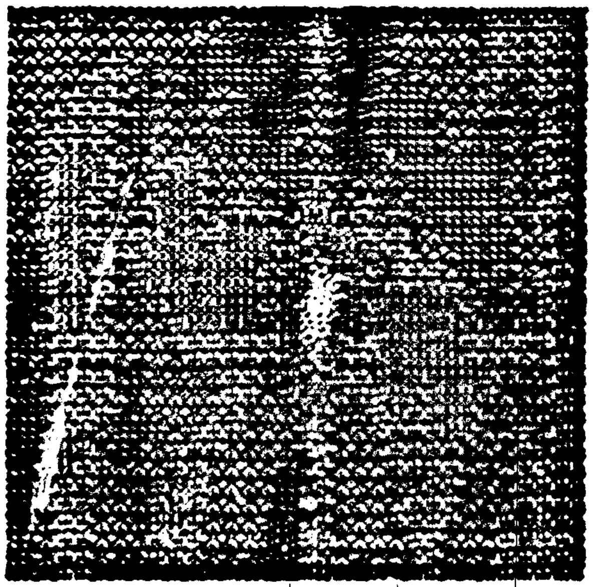


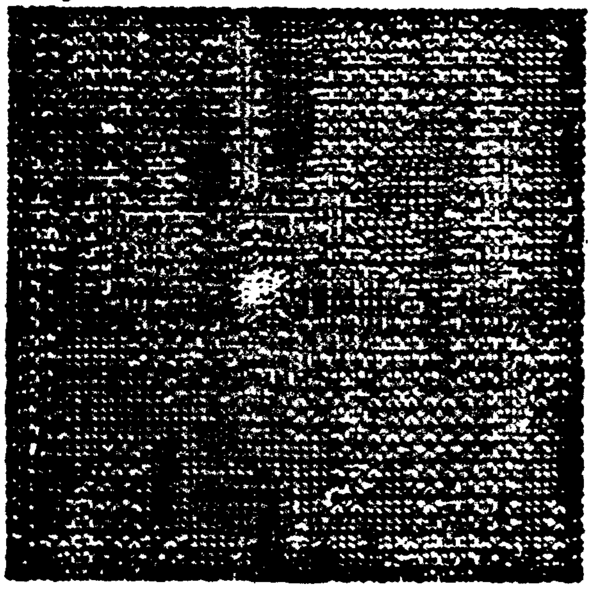
Figure 4.3 Some Files Before and After NORM.
a) Figure 4.2d, b) Normalization of Figure 4.2d,
c) Figure 4.2e, d) Normalization of Figure 4.2e
(No reduction of PIX output).



a

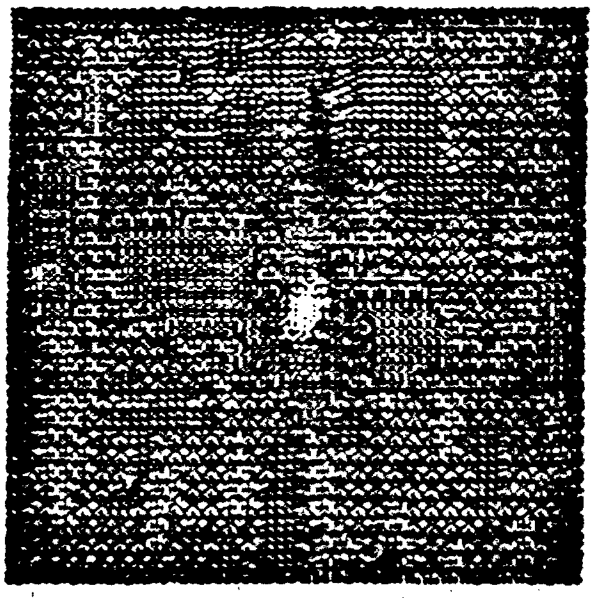


b



c

Figure 4.4a-c Second Pass Correlation Output.
a) Correlation Output from Figure 4.2a,
b) Correlation Output from Figure 4.2b,
c) Correlation Output from Figure 4.2c.
(No reduction of PIX output).


d

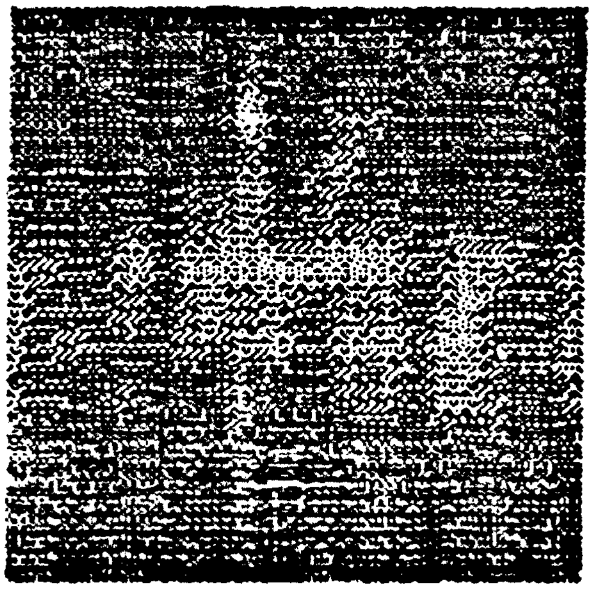

e

Figure 4.4d-e Second Pass Correlation Output.
d) Correlation Output from Figure 4.2d,
e) Correlation Output from Figure 4.2e.
(No reduction of PIX output)

V. EVALUATION

Visual inspection of the second pass correlation was confusing at times. While a sharp peak at the center of the file was clearly distinguishible as correlation with a target, less sharp peaks became uncertain as to their meaning. To improve the interpretation of the peaks a more detailed analysis of the nature of the files was used to produce the guidelines with would be used in the evaluation process.

5.1 THREE DIMENSIONAL REPRESENTATION

In order to increase perception of the information contained in the smaller output files of the recorrelation, a three-dimensional representation was sought. The Signal Processing Lab had access to software which produced a 3-D plot on either a Tektronix 4010 graphics display terminal or a Heathkit H-19 terminal with graphics capibility. With the 4010 an immediate printout was availible with a Tektronix 4631 hard copy unit.

The software package was called PLOT3D. It required that the information source for the plot be contained in a file of binary form. To obtain such a form from the standard packed video format, a program called BI was used to convert to binary form.

Figure 5.1 shows the 3-D representation of an autocorrelation performed on figure 4.4a. This shows what the output data should be in the best possible case for this type of image. Figure 5.2 shows the plot of Figure 4.4b, which is also considered a good correlation peak. Figure 5.3 is the 3-D plot of Figure 4.4e which is actually the correlation of the template file with a file containing noise and a partial target. One should note that this type of output contains a peak which is off centered and not as prominent as the peaks in the previous two diagrams.

5.2 HISTOGRAM ANALYSIS

Further analysis was done on these files by examining the histograms produced from them. The program HISTO produced the histogram of a given file. It provided the data in tabled form, displayed on a terminal, and in graphic form, displayed on a graphics terminal if one were availible.

Figure 5.4 is the histogram produced from the data of the auto-correlation of Figure 4.4a (shown 3-D in Figure 5.1). Since it was an auto-correlation, it was assumed that this is the optimum "peaking" for this type of image data. Figure 5.5 is the histogram of Figure 4.4b (shown 3-D in Figure 5.2). This is the output that can be expected from a good correlation match since the target was much like the template used. The histogram of Figure 5.6 is from Figure 4.4e (shown 3-D in Figure 5.3). This is from a correlation which is not judged to be a good match of target with template.

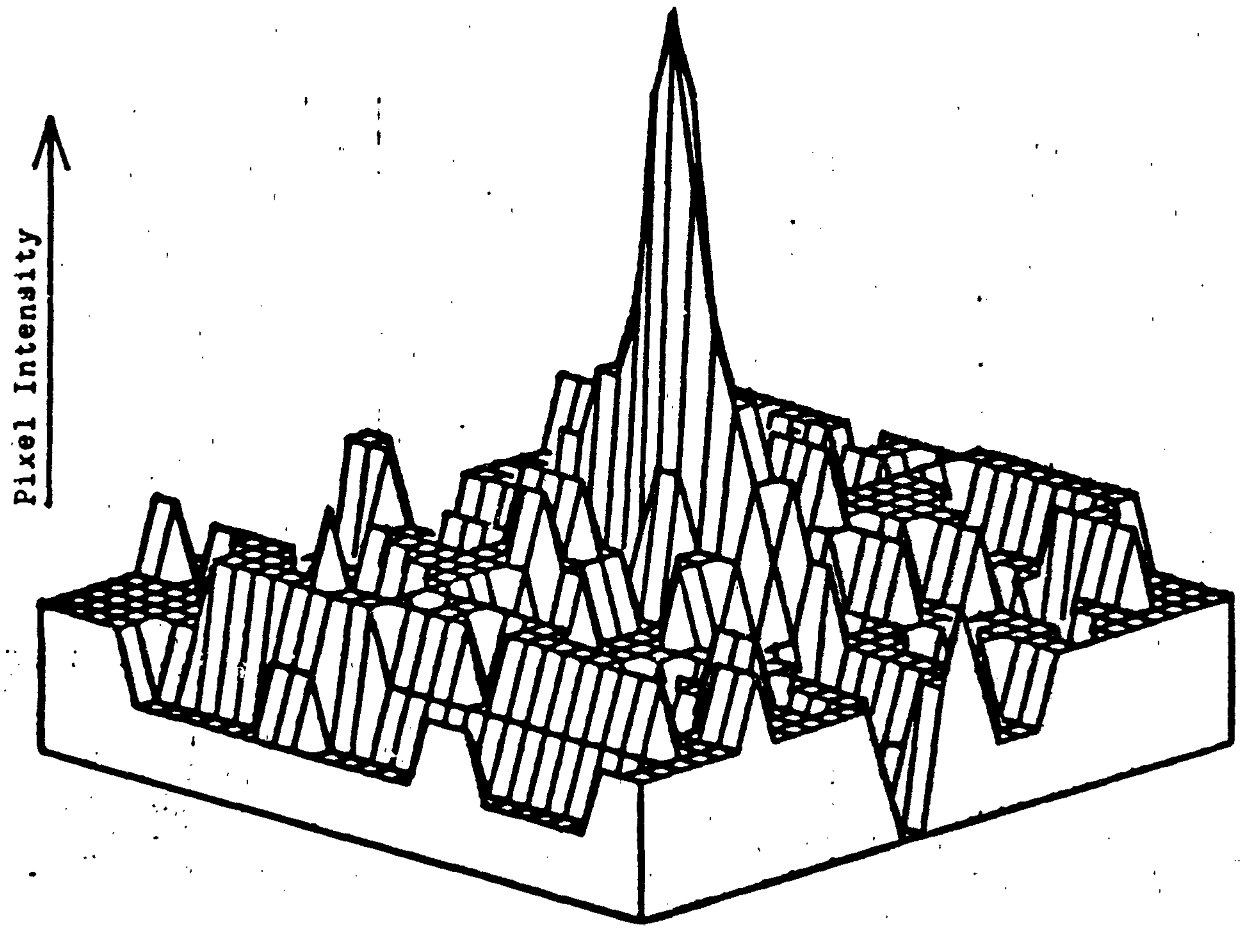


Figure 5.1 Three-dimensional Representation of Autocorrelation of Figure 4.4a.

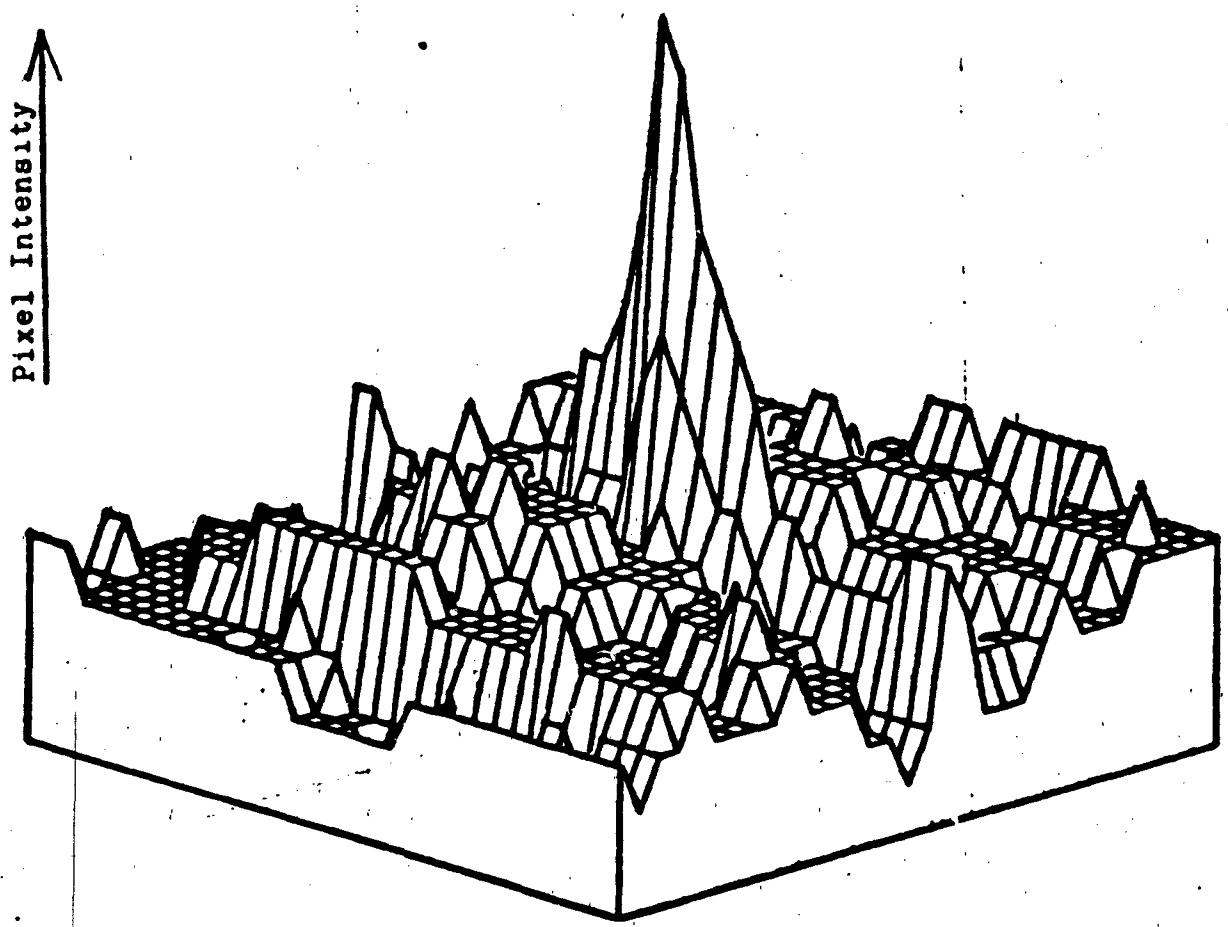


Figure 5.2 Three-dimensional Representation of Figure 4.4b.

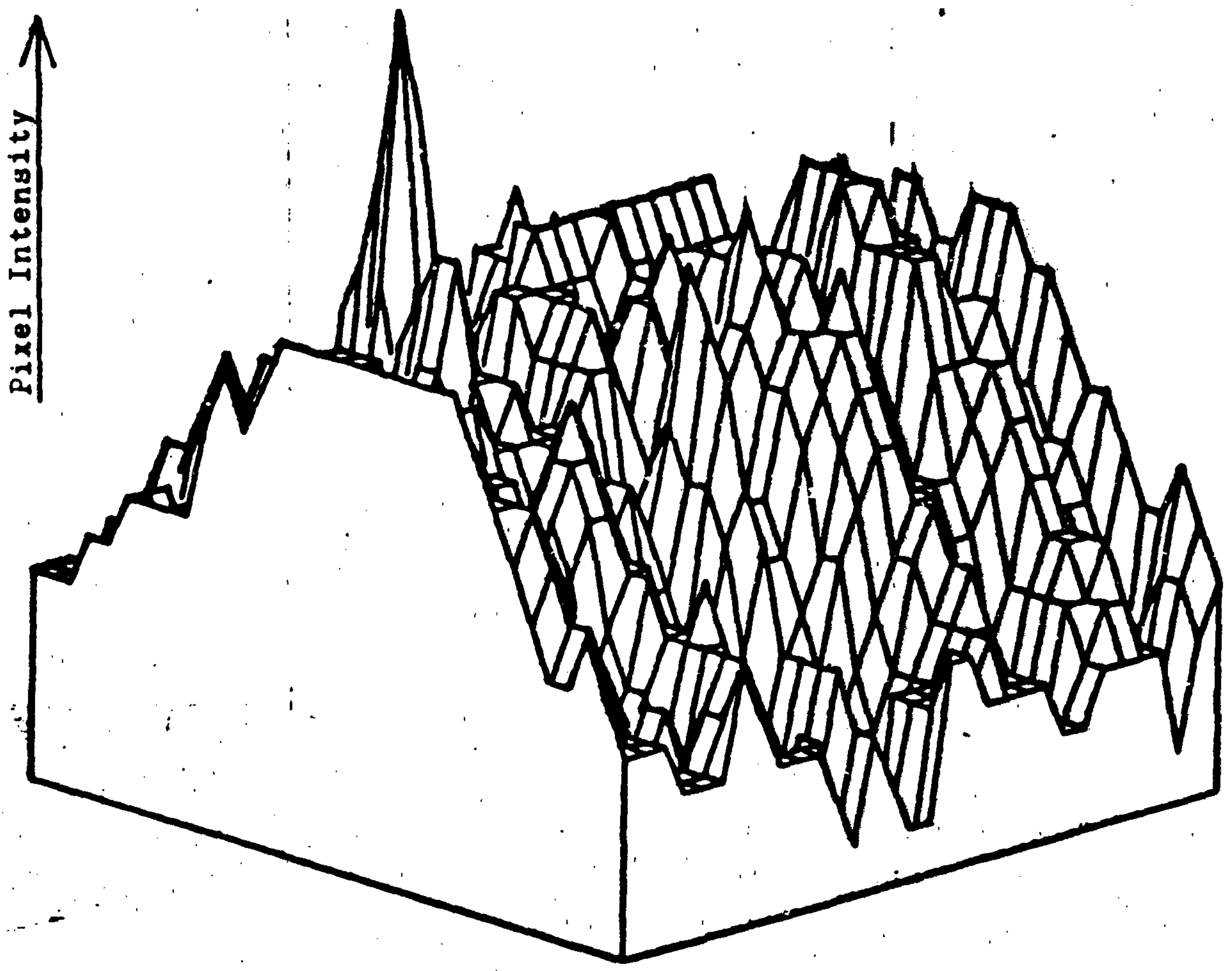


Figure 5.3 Three-dimensional Representation of Figure 4.4e.

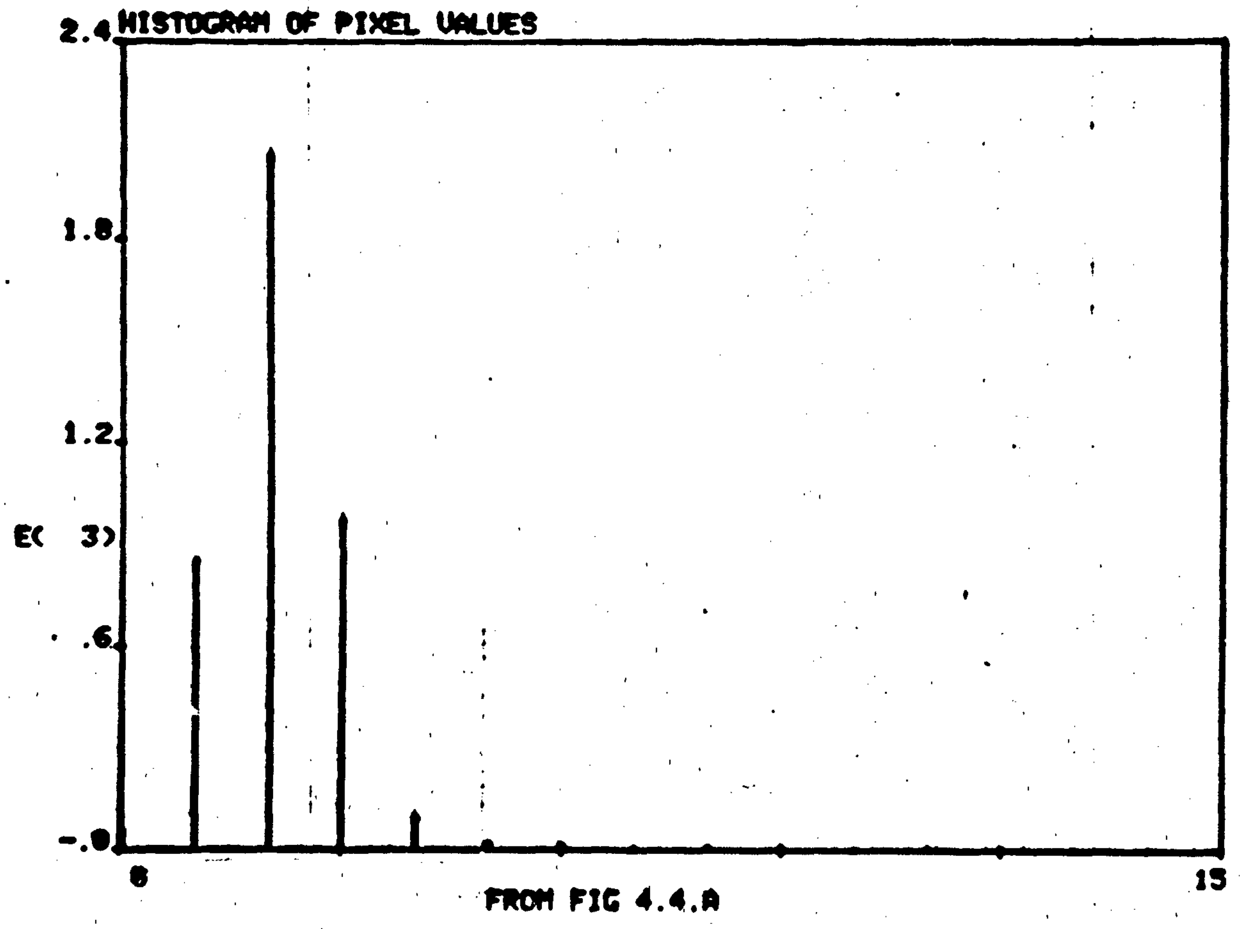


Figure 5.4 Histogram of Pixel Population of Figure 4.4a.

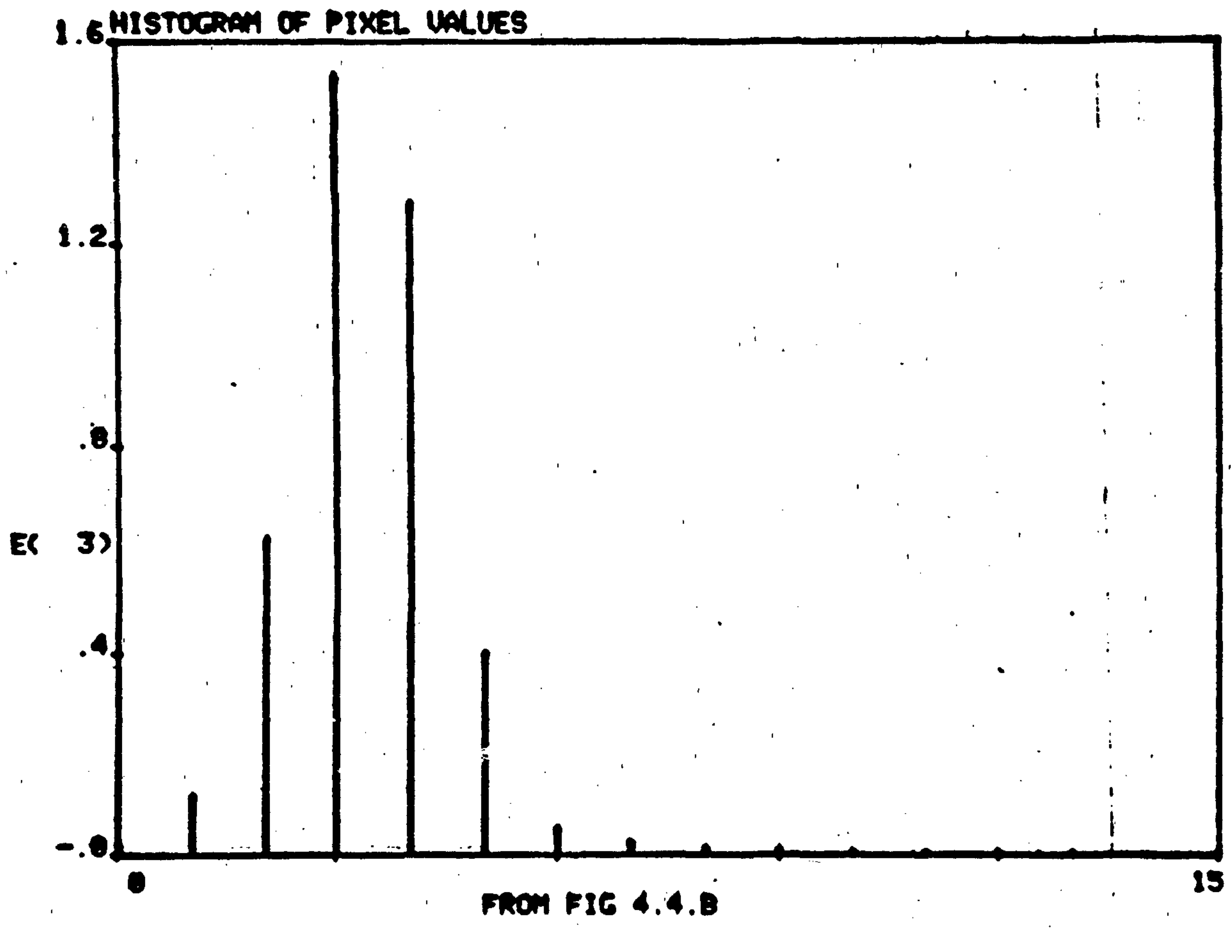


Figure 5.5 Histogram of Pixel Population of Figure 4.4b.

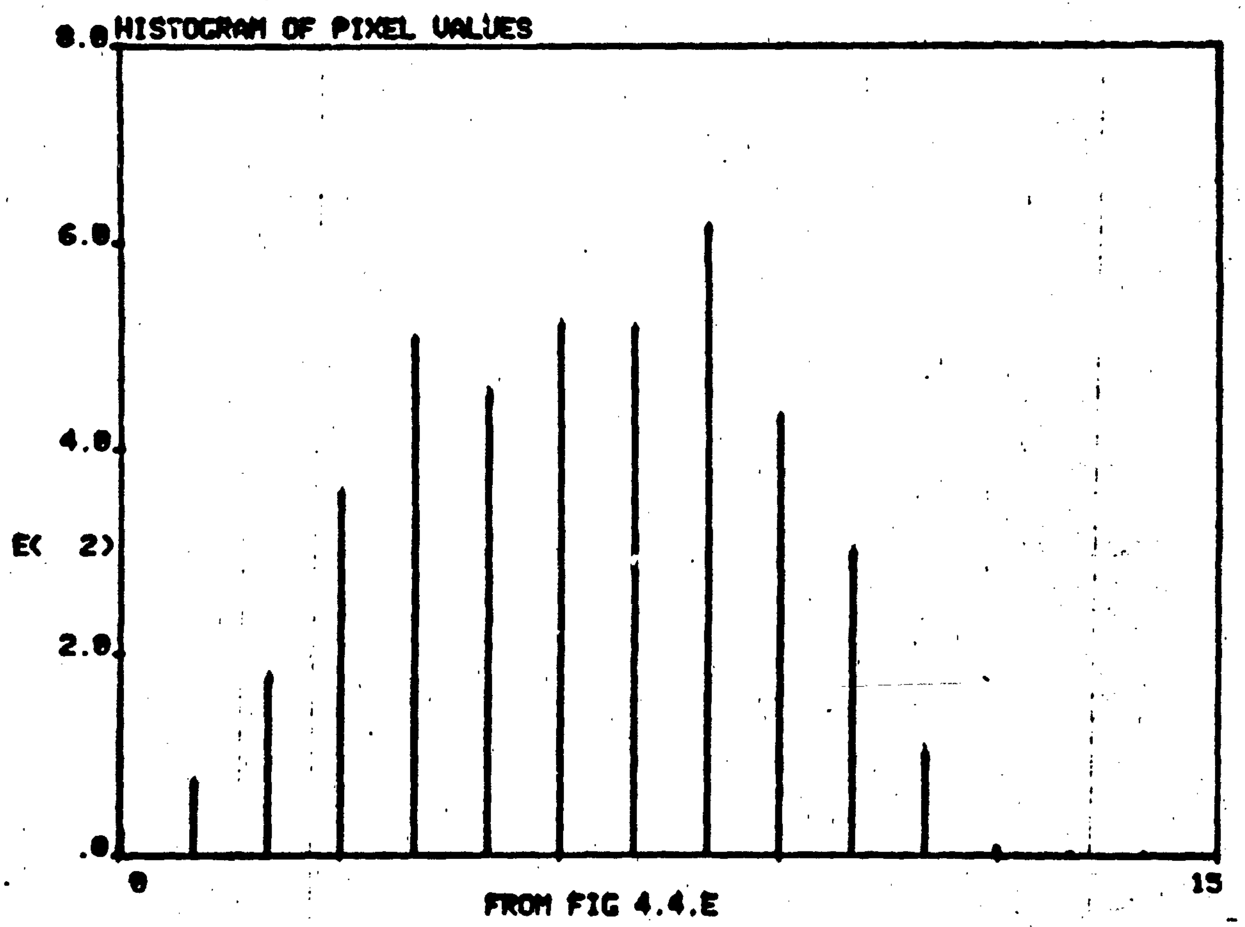


Figure 5.6 Histogram of Pixel Population of Figure 4.4e.

Examination of these three histograms indicated that a correlation which was considered to be good will produce a small number of pixels in the high intensity region; whereas, a correlation which was judged to be bad did not. A bad match seemed to produce a more even distribution of pixel population.

5.3 EVALUATION PROCESS

The recorrelated video files were evaluated by considering the two properties previously mentioned. This was accomplished with the program EVAL which produced two scores corresponding to these two properties.

The first score, Score1, rated distance of the location of the peak from the center of the file. A good correlation produces a peak very close to the file center, due to the initial alignment of the first pass correlation with the actual targets. Therefore, the indication of a good target-template match decreases as the distance to the peak from the center increases. The peak-to-center distance was normalized as follows:

$$N = \frac{D}{2F} \qquad (5\text{-}1)$$

where D is the peak-to-center distance, F is the file's pixel size and N is the normalized distance. This normalized distance was rated to produce Score1 with the function in Figure 5.7.

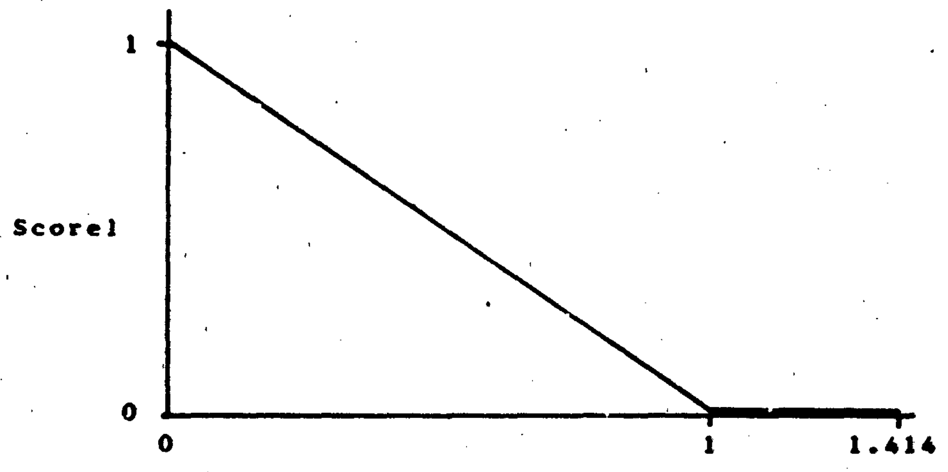


Figure 5.7 Function of the Score1 Evaluation.

The second score, Score2, rated the sharpness/uniqueness of the peak. A good correlation produces a single sharp peak. A measure of the peak's sharpness/uniqueness was done by a type of integration of pixel population within the histogram. The number of pixels in each intensity level was integrated from level 15 backwards to zero. When 5 percent of the total number of pixels was exceeded, the present intensity level was recorded. To that level was added the previous number pixels divided by the percent of the total number which was being sought.

This method produced a somewhat continuous output. This output (which varied from 1 for good, to 16 for bad) was used to produce Score2. The function which determined this score is given in equation 5-2. In this equation the output of the integration described is refered to as X. (X is the variable SPOT in the program EVAL.)

$$\text{SCORE2} = \begin{cases} 1.0 & ; \text{ for } X \le 6 \\ \sqrt{\dfrac{(16-X)}{10}} & ; \text{ for } 6 < X \le 16 \end{cases} \qquad (5\text{-}2)$$

The program which calculated these two scores, EVAL, also produces a third, total score refered to as SCORE. Score1 and Score2 are a rating from 0, for bad, to 1, for good. SCORE is a multiplication of these two factors (see equation 5-3). Due to this multiplication, any rating deficency in either the peak location or the peak sharpness/uniqueness will be reflected in the total rating, SCORE.

$$\text{SCORE} = (\text{SCORE1})\times(\text{SCORE2}) \qquad (5\text{-}3)$$

Table I is the hard copy output of this evaluation routine. This program also examines any inconsistency in the two scores. If the peak is sharp, but off centered, it is an indication of an partial off-centered target and is so stated in the output. Note that EVAL also ranks the final score from one to ten.

TABLE I

Evaluation of Second Pass Correlation Output Files

```
                     EVALUATION RESULTS
                SCENE FILENAME: FIGURE 4.4
FILE NAME        SCORE1        SCORE2     TOTAL SCORE      RANK
---------        ------        ------     -----------      ----
FIG404A.VD      .933709      1.000000       .933709         10
FIG404B.VD      .908891       .971016       .882548          9
FIG404C.VD      .889515       .961668       .855417          9
FIG404D.VD      .933709       .956321       .892926          9
FIG404E.VD      .374610       .666951       .249847          3
   POSSIBLE OFF-CENTERED PARTIAL TARGET
```

Note that the first 4 potential targets were rated high, while the fifth potential target, which was actually a partial target and high energy noise, was rated much lower.

5.4 RESULTS

The process described in this thesis seemed to work quite well with the images tested. Appendix E contains several of the additional images used for testing along with the results and evaluations.

As mentioned earlier in Chapter Four, the multiplying filter program, FVCM, was able to filter 64x64 sized files. However, it was not used in this capacity. The propose of the filter was to find specific targets with a general template. This was very useful during the first pass correlation process which purpose was to find the location of possible targets. It was the function of the second pass correlation process, using energy normalization, to re-examine the location after the potential targets were found. Trials showed that filtering during the second pass seemed to merely blur the correlation output as shown in Figure 5.8.

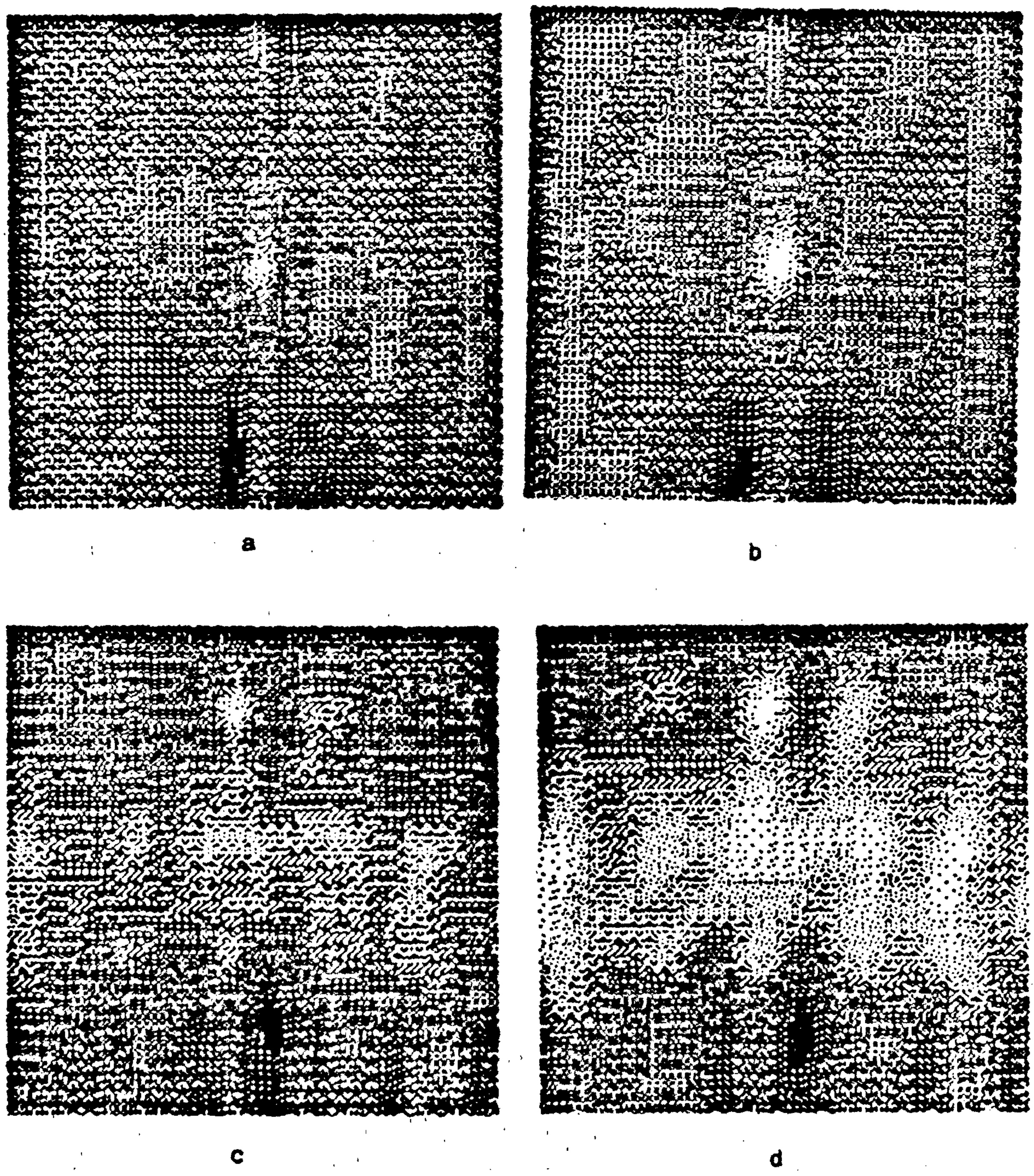


Figure 5.8 Second Pass Correlation Output Files, With and Without Filtering.
a) Figure 4.4a, b) Figure 4.4a with Filtering,
c) Figure 4.4e, d) Figure 4.4e with Filtering.
(No reduction of PIX output)

VI. DISCUSSION

6.1 CONCLUSIONS

The evaluation process discussed in Chapter Five produced tangible results which are included in Appendix E. In the cases where noise was not present, or low compared to the energy of the edges of the target, the process worked very well. In trials where general templates were used to find non-identical specific targets, the filtering routine proved to be effective.

There were examples of the process not being able to extract all of the targets out of the scene. One such case was the scene used throughout the text of this thesis. Although the template of the F-16 extracted from the scene (Figure 3.4) could find all the targets in the scene (Figure 2.7), the general template of Figure 3.6 was unable to locate the two F-16's at the outer edges of the scene. (This case is formally excluded with Assumption #4, Section 1.4.) Also, the aircraft just above the artifical noise was not as clear as the others. (See Figure 3.7)

The second pass correlation with energy normaliztion proved to be extremely helpful in eliminating the false target identification that occured due to high energy noise. Even in cases where target correlation seemed to be somewhat difficult, the program EVAL was able to make good evaluations between the relative outputs given (see Figure E-1 and Table E-II).

6.2 RECOMENDATIONS

Due to time constraints, the process developed in this thesis has had a finite amount of testing. Further testing may be done with higher background noise levels, less matching between target and template, as well as a combination of these two variations.

Another possible advancement with this algorithm could be its implementation with a master program which would direct the functions of all programs described. The visual inspection of correlation peaks after the first pass could be replaced with software techniques similar to those developed by Hamadani (Ref 2). Hamadani developed a blob detection technique which could be modified to automatically return the peak coordinates to a master program.

6.1 PROCESS SUMMARY

The process developed and described in this thesis is summarized in the flowchart of Figure 6.1. This flowchart illustrates the sequence of every step taken in this scene analysis process.

The actual file information is described in normal type. The program which produces the information is in capital letters and any possible substitutuions are after the slash. The optional processing is in square brackets.

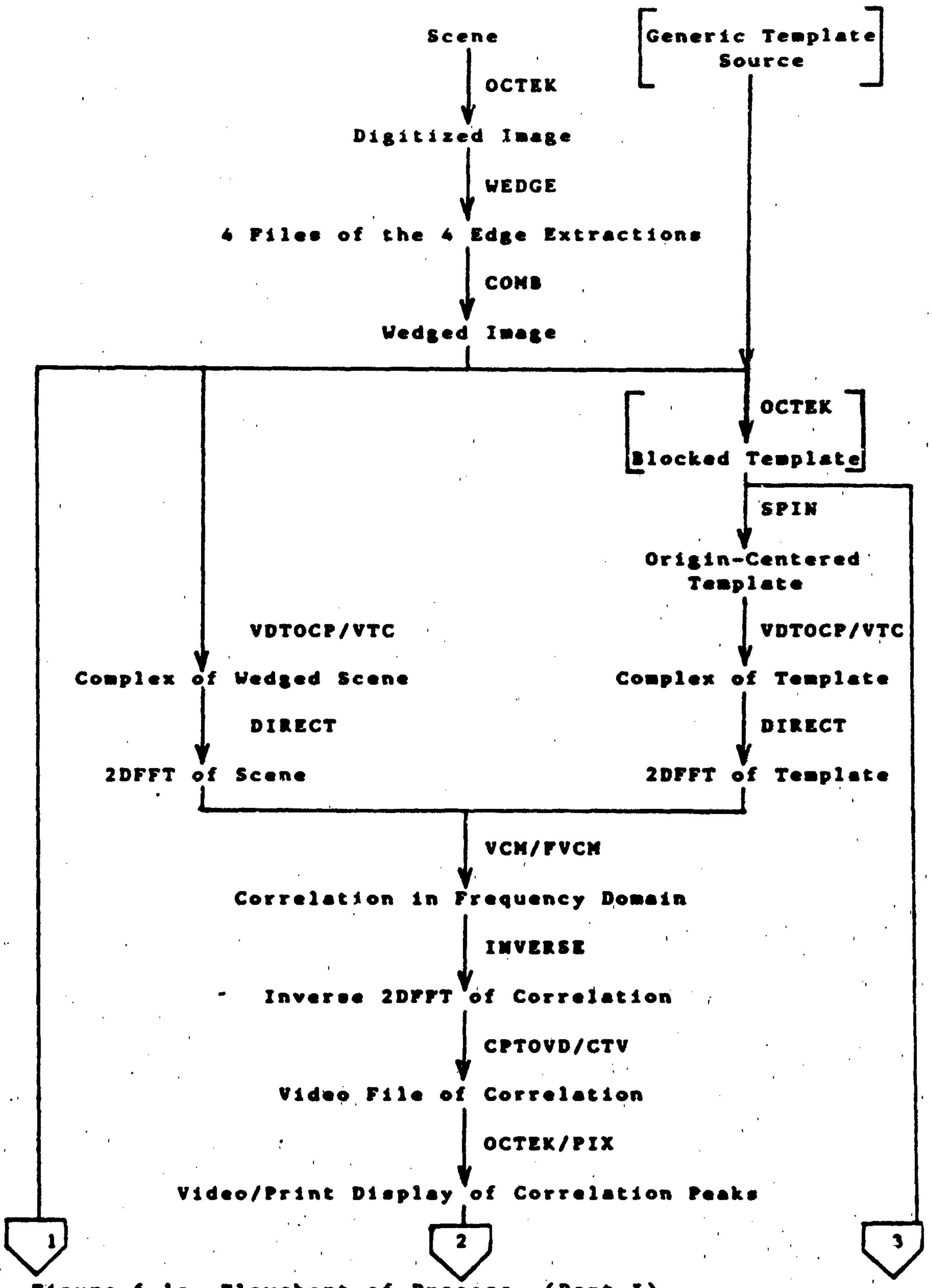


Figure 6.1a Flowchart of Process (Part I)

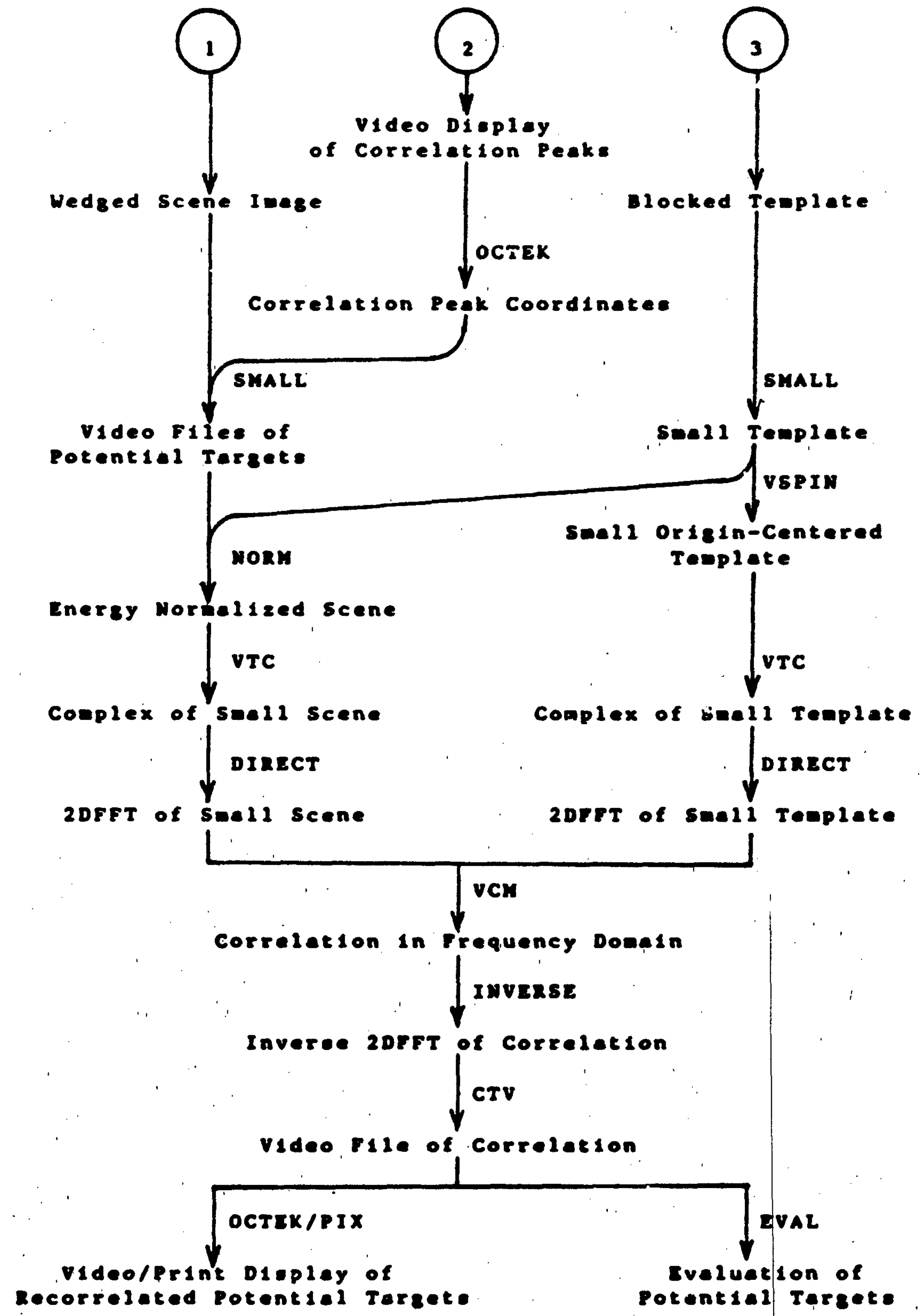

1
2
3
Video Display of Correlation Peaks
Wedged Scene Image
Blocked Template
OCTEK
Correlation Peak Coordinates
SMALL
SMALL
Video Files of Potential Targets
Small Template
VSPIN
Small Origin-Centered Template
NORM
Energy Normalized Scene
VTC
VTC
Complex of Small Scene
Complex of Small Template
DIRECT
DIRECT
2DFFT of Small Scene
2DFFT of Small Template
VCM
Correlation in Frequency Domain
INVERSE
Inverse 2DFFT of Correlation
CTV
Video File of Correlation
OCTEK/PIX
EVAL
Video/Print Display of Recorrelated Potential Targets
Evaluation of Potential Targets

Figure 6.1b Flowchart of Process (Part II)

BIBLIOGRAPHY

1. Cromer, 2Lt James H. Scene Analysis: Non-linerar Spatial Filtering For Automatic Target Detection. MS thesis GE/EE/82D-26. School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, December 1982. (AD A124 788)

2. Hamadani, Naser A. Automatic Target Cueing In IR Imagery. MS thesis GEO/EE/81D-3. School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, December 1981. (AD A115 480)

3. Horev, Maj Moshe. Picture Correlation Model for Automatic Machine Recognition. MS thesis GE/EE/80D-25. School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB, OH, December 1980. (AD A100 765)

4. Kabrisky, Dr. M., Professor, Air Force Institute of Technology, Wright-Patterson AFB, OH, Personal interviews. July-Oct 1984.

5. King, Capt. David A., Programmer's Guide - PICBUF - A Virtual Memory System for Digital Picture Processing. Signal Processing Laboratory, Air Force Institute of Technology, Wright-Patterson AFB, Ohio, 5 July 1984.

6. McGillem. Clare D. and Cooper, George R. Continuous and Discrete Signal and System Analysis. New York: Holt, Rinehart and Wilson, Inc. 1974.

7. McKeeman, John C., Two-Dimensional Recognition Using A Modified AFIT Edge Enhancement Algorithm. PhD dissertation, University of Dayton, Dayton, Ohio. 1984.

8. Rosenfeld, A. "Image Pattern Recognition," Proceedings of the IEEE, 69:5. pp. 596-605, May 1981.

9. Simmons, R.A. Machine Segmentation of Unformatted Characters. MS thesis GE/EE/81D-54. School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB, OH, December 1981. (AD A115 556)

10. Wells, Capt Robert D. Image Filtering with Boolean and Statisical Operators. MS thesis GE/EE/83D-72. School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB, OH, December 1983. (AD A138 421)

APPENDIX A

Materials and Equipment

HARDWARE

The Signal Processing Lab at the Air Force Institute of Technology at Wright-Patterson AFB, Ohio had all the necessary hardware and support needed for this thesis. The lab has a Data General Nova 2 computer and a Data General Eclipse S/250 computer as well as several hard disks for mass storage. All necessary terminals and printers were available.

The image input/output was all done through the Data General Nova computer. The image processing station in the lab contains an Octek 2000 Image Analyzer, a Dage 650 video camera, an Electrohome video monitor and a Tektronix 4632 video hard copy unit.

The graphics station provided two Tektronix 4010 graphics terminals, controlled by the Data General Eclipse computer and a Tektronix 4631 hard copy unit which was connected to both graphics terminals.

SOFTWARE

The Signal Processing Lab had access to several software packages which were extensivly used in this thesis. The following pages are explanations on the use of the software for which the source code was unavaible (Direct, Inverse, and Plot3D) and for OCTEK which was drove the Octek

2000 Image Analyzer. Also included are notes on PICBUF, the virtual memory system for digital picture processing, and PIX, the image printing program.

Notes on the program DIRECT

Use: Two Dimensional Discrete Fourier Transform (2DFFT)

Execution format:

```
        DIRECT inputfile/I #/N [outputfile/O]
```

Parameter explanation:

Inputfile: The input file must be in binary format since a RDBLK is used to read data from the input file. The input file is assumed to contain complex data, with the imaginary part zero. The input array is assumed to be square, with a minimum row or column length of 64. If an array of size < 64 is to be used as input, each column should be stored as one block of data even though only the first N complex numbers in the block correspond to actual data points. Only N rows need be stored. Thus, for N < 64, the size of the input array must be 64 by N columns.

#: An integer must be specified to indicate the size of the input array (NxN), which is the size of the DFT computed. N must be a power of two, N < 512.

Outputfile: The output file specification is optional. If no output file is specified, the transformed data will be written back into the input file, destroying the input data. The output file is complex and in binary format. The origin of the DFT samples will be at the location (N/2, N/2), and the rows and columns will be transposed. Note: This program runs much faster if the outputfile is not specified. (Ref 8)

Notes on the program INVERSE

Use: Two Dimensional Inverse Discrete Fourier Transform

Execution format:

```
INVERSE inputfile/I [outputfile/O] #/N
```

Parameter explanation:

__Inputfile:__ The input file must be in binary format since a RDBLK is used to read data from the input file. The input file is assumed to contain complex data, and must have minimum length of 64. If the NxN array has N < 64, the columns should be augmented with zeros so that each column can be stored as one block of data. Thus, for N < 64, the size of the input array must be 64 rows by N columns of complex DFT samples. The origin of the frequency axes must be at (N/2 + 1, N/2 + 1).

__Outputfile:__ The output file specification is optional. If no output file is specified, the inverse-transformed data will be written back into the input file, destroying the input data. The output file is complex and in binary format. Note: This program runs much faster if the outputfile is not specified.

__#:__ An integer must be specified to indicate the size of the input array (NxN), which is the size of the DFT computed. N must be a power of two, N < 512. (Ref 8)

Notes on program PLOT3D

Use: Three-Dimensional Representation of a Binary File

Execution format:

```
PLOT3D inputfile/I
```

Explaination:

Inputfile: The input file to this program must be written in binary format. This is accomplished on video files with the program BI.

This program must be run on a terminal with graphics capability, such as a Tektronix 4010 or an H-19 with a graphics package. A hard copy is availible via a Tektronix 4631 hardcopier. The inputs prompts are self-explanitory with the exception of the following:

XSIZE: Relative base size in the X direction on the screen (typically 8).

YSIZE: Relative base size in the Y direction on the screen (typically 3).

HEIGHT: Relative size in the Z direction (typically 5).

NORMALIZATION: Yes, the output of BI needs to be normalized.

lotes on the program OCTEK

lse: Operation of the Octek 2000 Image Analyzer

:xecution format:

```
        OCTEK
```

xplanation:

he following menu is the available options in the program CTEK:

```
 1.  ACQUIRE   - take picture (with TV camera #1)
 2.  LOAD      - display video (.VD) file
 3.  SAVE      - save display in video (.VD) file
 4.  CURSOR    - activate cursor ot display pixel values
 5.  CLEAR     - clears display
 6.  BARS      - display greyscale bar test pattern
11.  ACQUIRE2  - take picture (with TV camera #2)
12.  BINARIZE  - threshold display
13.  LOOKUP    - alter output translation table
14.  NEGATIVE  - display negative
15.  GREYSCALE - restore true greyscale
16.  FILL      - fill block
-1.  QUIT      - exit program
```

TEK options 1, 2 and 3 were used for input and output of l video files. Option 4, CURSOR, was used to determine e location of correlation peaks. Option 12, BINARIZE, was ed to anaylize correlation output files. Option 16, FILL, s used to eliminate unwanted noise in a template video le.

Notes on libary PICBUF

Use: Virtual Memory System for Digital Picture Processing

Reloader reference:

F5PICBUF.LB (for Fortran V on the Eclipse)

NF4PICBUF.LB (for Fortran IV on the Nova)

Explanation:

PICBUF is a libary of Fortran subroutines which implements a virtual memory system capable of storing and accessing two-dimentional digital data. The PICBUF routines create virtual memory buffers, accesse image data rows and pixel points, and store the processed buffers into disk files. An extended explanation of these routines is included in Reference 5.

The PICBUF libary was used in the following programs and subroutines:

SPIN.FR

MOVE.FR

SMALL.FR

VSPIN.FR

VID.FR

WINDOW.FR

EVAL.FR

Notes on program PIX

Use: Hardcopy output of video file.

Execution format:

```
        PIX
```

Explanation:

This program will convert video pixels to lineprinter pixels, and will put the picture in a file or to the Printronix 300 lineprinter. This program prints either a complete 256x256 pixel picture, or a smaller picture that is of a desired length specified by the user.

This program was originally written by Lt. Simmons, revised by Lt. Cromer and later by Jim Wetz. (Ref 1,9)

APPENDIX B

Preprocessing Software

This Appendix contains the prepocessing software used to perform edge extraction on the video files. Included are the following programs and subroutines:

WEDGE.FR

WROW1.FR

COMB.FR

```
C*****************************************************************
C                                                                *
C  WEDGE.FR - DG FORTRAN 5 - CAPT ROBERT WELLS, DEC 1983         *
C                   REDONE BY LT RICHARD MILLS JULY 1984         *
C                                                                *
C     This program edges the input file (packed video format)    *
C     with  a mask operator called the Wedge Operator. The       *
C     output is separated into four 256 X 256 packed video       *
C     files according to which mask produced the maximum value.  *
C     If two masks of different orientations produce the         *
C     maximum, that value goes to both output files.             *
C                                                                *
C     RELOAD COMMAND:                                            *
C       RLDR WEDGE UNPACK REPACK WROW1 @FLIB@                    *
C                                                                *
C     EXECUTION COMMAND:                                         *
C       WEDGE                                                    *
C                                                                *
C*****************************************************************

      INTEGER INMAT(256,16),TEMP(512),TOGGLE
      INTEGER OMAT1(256,8),OMAT2(256,8),OMAT3(256,8),OMAT4(256,8)
      INTEGER INFLNM(7),OFLNM1(7),OFLNM2(7),OFLNM3(7),OFLNM4(7)
C
C ACCEPT INPUT
C
      ACCEPT"ENTER INPUT FILENAME ->"
      READ(11,1)INFLNM(1)
    1 FORMAT(S13)

      ACCEPT"ENTER OUTPUT FILENAME #1 ->"
      READ(11,1)OFLNM1(1)

      ACCEPT"ENTER OUTPUT FILENAME #2 ->"
      READ(11,1)OFLNM2(1)

      ACCEPT"ENTER OUTPUT FILENAME #3 ->"
      READ(11,1)OFLNM3(1)

      ACCEPT"ENTER OUTPUT FILENAME #4 ->"
      READ(11,1)OFLNM4(1)

      CALL OPEN(1,INFLNM,1,IER)
      CALL CHECK(IER)
C
C DELETE OUTPUT FILES IF THEY EXIST AND (RE)CREATE THEM
C
      CALL DFILW (OFLNM1,IER)
      CALL CFILW (OFLNM1,2,KER)
      CALL CHECK(KER)
      OPEN 2, OFLNM1,ATT="OR",ERR=100
      CALL DFILW (OFLNM2,IER)
      CALL CFILW (OFLNM2,2,KER)
      CALL CHECK(KER)
      OPEN 3, OFLNM2,ATT="OR",ERR=100

      CALL DFILW (OFLNM3,IER)
      CALL CFILW (OFLNM3,2,KER)
      CALL CHECK(KER)
      OPEN 4, OFLNM3,ATT="OR",ERR=100

      CALL DFILW (OFLNM4,IER)
      CALL CFILW (OFLNM4,2,KER)
      CALL CHECK(KER)
      OPEN 5, OFLNM4,ATT="OR",ERR=100

100   DO 2 I=1,256     ; FILL FIRST AND LAST 4 ROWS OF
         TEMP(I)=0     ; OF OUTPUT FILES WITH ZERO
    2 CONTINUE
      CALL WRBLK(2,0,TEMP,1,IER)
      CALL CHECK(IER)
      CALL WRBLK(2,63,TEMP,1,IER)
      CALL CHECK(IER)
      CALL WRBLK(3,0,TEMP,1,IER)
      CALL CHECK(IER)
      CALL WRBLK(3,63,TEMP,1,IER)
      CALL CHECK(IER)
      CALL WRBLK(4,0,TEMP,1,IER)
      CALL CHECK(IER)
      CALL WRBLK(4,63,TEMP,1,IER)
      CALL CHECK(IER)
      CALL WRBLK(5,0,TEMP,1,IER)
      CALL CHECK(IER)
      CALL WRBLK(5,63,TEMP,1,IER)
      CALL CHECK(IER)

      CALL RDBLK(1,0,TEMP,2,IER) ;READ IN FIRST 8 ROWS
      CALL CHECK(IER)
      CALL UNPACK(512,TEMP,INMAT)

      TOGGLE=0  ; FLAG TO SHOW BUFFER WRAP-AROUND

      DO 5 I=1,31 ; COUNTER FOR 2K BUFFER LOADS

         WRITE FREE(10)I          ;Keep track of running program

         CALL RDBLK(1,2*I,TEMP,2,IER) ;READ NEXT 8 ROWS
         CALL CHECK(IER)
         CALL UNPACK(512,TEMP,INMAT(1,9-8*TOGGLE))

         DO 3 K=1,8 ; COUNTER FOR OUTPUT BUFFER ROWS
            CALL WROW1(K,TOGGLE,INMAT,OMAT1,OMAT2,OMAT3,OMAT4)
    3    CONTINUE
```

```
      CALL REPACK(512,OMAT1,TEMP)
      CALL WRBLK(2,2*I-1,TEMP,2,IER) ; WRITE 8 ROWS OF RESULTS
      CALL CHECK(IER)
      CALL REPACK(512,OMAT2,TEMP)
      CALL WRBLK(3,2*I-1,TEMP,2,IER)
      CALL CHECK(IER)
      CALL REPACK(512,OMAT3,TEMP)
      CALL WRBLK(4,2*I-1,TEMP,2,IER)
      CALL CHECK(IER)
      CALL REPACK(512,OMAT4,TEMP)
      CALL WRBLK(5,2*I-1,TEMP,2,IER)
      CALL CHECK(IER)

      IF(TOGGLE.EQ.1)GO TO 4
         TOGGLE=1     ; TOGGLE THE BUFFER WRAP-AROUND FLAG
         GO TO 5
4        TOGGLE=0

5 CONTINUE

  CALL RESET
  STOP"<7><7><7><7>WEDGE"
  END
```

```
      SUBROUTINE WROW1(K,TOGGLE,INMAT,OMAT1,OMAT2,OMAT3,OMAT4)

C     BY WELLS 1983

      INTEGER K,TOGGLE,INMAT(256,16),MEDGE
      INTEGER OMAT1(256,8),OMAT2(256,8),OMAT3(256,8),OMAT4(256,8)
      REAL A1,A2,A3,A4,B1,B2,B3,B4,C1,C2,C3,C4,D1,D2,D3,D4,MAXIM

      IF(TOGGLE.EQ.1)GO TO 1  ; FIND INPUT BUFFER INDEXES
         K1=K+3
         K2=K+4
         K3=K+5
         GO TO 2
1        K1=MOD(K+10,16)+1
         K2=MOD(K+11,16)+1
         K3=MOD(K+12,16)+1
2     CONTINUE

      DO 3 J=1,4      ; ZERO FILL FIRST/LAST 4 COLUMNS OF RESULTS
         OMAT1(J,K)=0
         OMAT1(J+252,K)=0
         OMAT2(J,K)=0
         OMAT2(J+252,K)=0
         OMAT3(J,K)=0
         OMAT3(J+252,K)=0
         OMAT4(J,K)=0
         OMAT4(J+252,K)=0
3     CONTINUE

      DO 4 J=5,252  ; EDGE THESE COLUMNS

         J1=J-1
         J2=J
         J3=J+1

         A1=0.600*(INMAT(J1,K1)+INMAT(J2,K1)+INMAT(J3,K1))-
     +      0.300*(INMAT(J1,K2)+INMAT(J2,K2)+INMAT(J3,K2)+
     +             INMAT(J1,K3)+INMAT(J2,K3)+INMAT(J3,K3))

         A2=0.600*(INMAT(J1,K3)+INMAT(J2,K3)+INMAT(J3,K3))-
     +      0.300*(INMAT(J1,K2)+INMAT(J2,K2)+INMAT(J3,K2)+
     +             INMAT(J1,K1)+INMAT(J2,K1)+INMAT(J3,K1))

         A3=-A1

         A4=-A2

         B1=0.600*(INMAT(J2,K1)+INMAT(J3,K1)+INMAT(J3,K2))-
     +      0.300*(INMAT(J1,K1)+INMAT(J2,K2)+INMAT(J3,K3)+
     +             INMAT(J1,K2)+INMAT(J1,K3)+INMAT(J2,K3))
```

```
      B2=0.600*(INMAT(J1,K2)+INMAT(J1,K3)+INMAT(J2,K3))-
     +   0.300*(INMAT(J1,K1)+INMAT(J2,K2)+INMAT(J3,K3)+
     +          INMAT(J2,K1)+INMAT(J3,K1)+INMAT(J3,K2))
      B3=-B1
      B4=-B2

      C1=0.600*(INMAT(J3,K1)+INMAT(J3,K2)+INMAT(J3,K3))-
     +   0.300*(INMAT(J2,K1)+INMAT(J2,K2)+INMAT(J2,K3)+
     +          INMAT(J1,K1)+INMAT(J1,K2)+INMAT(J1,K3))

      C2=0.600*(INMAT(J1,K1)+INMAT(J1,K2)+INMAT(J1,K3))-
     +   0.300*(INMAT(J2,K1)+INMAT(J2,K2)+INMAT(J2,K3)+
     +          INMAT(J3,K1)+INMAT(J3,K2)+INMAT(J3,K3))

      C3=-C1

      C4=-C2

      D1=0.600*(INMAT(J1,K2)+INMAT(J1,K1)+INMAT(J2,K1))-
     +   0.300*(INMAT(J1,K3)+INMAT(J2,K2)+INMAT(J3,K1)+
     +          INMAT(J2,K3)+INMAT(J3,K3)+INMAT(J3,K2))

      D2=0.600*(INMAT(J2,K3)+INMAT(J3,K3)+INMAT(J3,K2))-
     +   0.300*(INMAT(J1,K3)+INMAT(J2,K2)+INMAT(J3,K1)+
     +          INMAT(J1,K2)+INMAT(J1,K1)+INMAT(J2,K1))

      D3=-D1

      D4=-D2

      MAXIM=AMAX1(A1,A2,A3,A4,B1,B2,B3,B4,C1,C2,C3,C4,D1,D2,D3
     + ,D4)
      OMAT1(J,K)=0
      OMAT2(J,K)=0
      OMAT3(J,K)=0
      OMAT4(J,K)=0

      MEDGE=ANINT(MAXIM)       ; INTEGRAL EDGE VALUE

      IF(MEDGE.GT.15)MEDGE=15  ; TEST FOR OVERFLOW
      IF(MEDGE.LT.0)MEDGE=0    ; AND UNDERFLOW

      IF(A1.EQ.MAXIM.OR.A2.EQ.MAXIM.OR.A3.EQ.MAXIM.OR.A4.EQ.
     +   MAXIM)OMAT1(J,K)=MEDGE    ; ORIENTATION #1

      IF(B1.EQ.MAXIM.OR.B2.EQ.MAXIM.OR.B3.EQ.MAXIM.OR.B4.EQ.
     +  MAXIM)OMAT2(J,K)=MEDGE    ; ORIENTATION #2

      IF(C1.EQ.MAXIM.OR.C2.EQ.MAXIM.OR.C3.EQ.MAXIM.OR.C4.EQ.
     +  MAXIM)OMAT3(J,K)=MEDGE    ; ORIENTATION #3
```

```
      IF(D1.EQ.MAXIM.OR.D2.EQ.MAXIM.OR.D3.EQ.MAXIM.OR.D4.EQ.
     +  MAXIM)OMAT4(J,K)=MEDGE    ; ORIENTATION #4

    4 CONTINUE

      RETURN
      END
```

```
C     PROGRAM COMB -
             DG FORTRAN 5 - BY CAPT ROBERT WELLS, DEC 1983

C    THIS PROGRAM COMBINES UP TO FOUR VIDEO IMAGES (PACKED VIDEO
C    FORMAT). EACH PIXEL OF THE INPUT IMAGES IS TESTED TO SEE IF
C    IT IS NON-ZERO. THE VALUE OF THE FIRST NON-ZERO PIXEL OF THE
C    INPUT IMAGES IS ASSIGNED TO THE OUTPUT IMAGE AT THAT PIXEL
C    LOCATION. IF ALL INPUT IMAGES ARE ZERO, THE OUTPUT IS ZERO.
C    THE OUTPUT IS STORED IN PACKED VIDEO FORMAT.

      INTEGER INFNM1(7),INFNM2(7),INFNM3(7),INFNM4(7),OUTFNM(7)
      INTEGER IN1(2048),IN2(2048),IN3(2048),IN4(2048),OUT(2048)
      INTEGER TEMP(512),FLCNT

    1 FORMAT(S13)
    2 ACCEPT"ENTER NUMBER OF FILES TO BE COMBINED ->",FLCNT
      IF((FLCNT.LT.2).OR.(FLCNT.GT.4))GO TO 2

      GO TO (2,4,3) FLCNT

      ACCEPT"ENTER INPUT FILENAME ->"
      READ(11,1)INFNM4(1)
      CALL OPEN(5,INFNM4,1,IER)
      CALL CHECK(IER)

    3 ACCEPT"ENTER INPUT FILENAME ->"
      READ(11,1)INFNM3(1)
      CALL OPEN(4,INFNM3,1,IER)
      CALL CHECK(IER)

    4 ACCEPT"ENTER INPUT FILENAME ->"
      READ(11,1)INFNM2(1)
      CALL OPEN(3,INFNM2,1,IER)
      CALL CHECK(IER)

      ACCEPT"ENTER INPUT FILENAME ->"
      READ(11,1)INFNM1(1)
      CALL OPEN(2,INFNM1,1,IER)
      CALL CHECK(IER)

      ACCEPT"ENTER OUTPUT FILENAME ->"
      READ(11,1)OUTFNM(1)
      CALL OPEN(1,OUTFNM,3,IER)
      CALL CHECK(IER)

      DO 14 I=0,31   ; COUNT FOR 2K BUFFER LOADS

         GO TO (2,6,5) FLCNT

         CALL RDBLK(5,2*I,TEMP,2,IER)
         CALL CHECK(IER)
         CALL UNPACK(512,TEMP,IN4)
```

```
 5     CALL RDBLK(4,2*I,TEMP,2,IER)
       CALL CHECK(IER)
       CALL UNPACK(512,TEMP,IN3)

 6     CALL RDBLK(3,2*I,TEMP,2,IER)
       CALL CHECK(IER)
       CALL UNPACK(512,TEMP,IN2)

       CALL RDBLK(2,2*I,TEMP,2,IER)
       CALL CHECK(IER)
       CALL UNPACK(512,TEMP,IN1)

       DO 13 J=1,2048 ; COMBINING LOOP

          GO TO (2,8,7) FLCNT

          IF(IN4(J).NE.0)GO TO 9

 7        IF(IN3(J).NE.0)GO TO 10

 8        IF(IN2(J).NE.0)GO TO 11

          IF(IN1(J).NE.0)GO TO 12

          OUT(J)=0 ;ZERO IF ALL IN-FILES ARE ZERO
          GO TO 13

 9        OUT(J)=IN4(J)
          GO TO 13

10        OUT(J)=IN3(J)
          GO TO 13

11        OUT(J)=IN2(J)
          GO TO 13

12        OUT(J)=IN1(J)

13     CONTINUE

       CALL REPACK(512,OUT,TEMP)
       CALL WRBLK(1,2*I,TEMP,2,IER)
       CALL CHECK(IER)

14 CONTINUE

   CALL RESET
   STOP"<7><7><7><7>COMB"
   END
```

APPENDIX C

Correlation, Normalization, and Evalution Software

This Appendix contains the main software developed and used in this thesis. Included are the following programs and subroutines:

SPIN.FR

MOVE.FR

VDTOCP.FR

VCM.FR

FVCM.FR

CPTOVD.FR

SMALL.FR

VSPIN.FR

NORM.FR

VTC.FR

CTV.FR

EVAL.FR

```
C*********************************************************************
C                                                                    *
C SPIN.FR -DG FORTRAN 5- LT RICHARD L. MIL       27 JULY 1984        *
C                                                                    *
C   This program will accept the name of a file which contains       *
C   a template and create a new file which places the template       *
C   center at the orgin of the video file.  The only other info      *
C   needed is the X-Y coordinates (Row-Column) of the original       *
C   template's position.  The background of the new file will        *
C   be set to black (pixel value = 0).                               *
C                                                                    *
C    RELOAD LINE:                                                    *
C      RLDR SPIN MOVE F5PICBUF.LB @FLIB@                             *
C                                                                    *
C    COMMAND LINE:                                                   *
C      SPIN                                                          *
C                                                                    *
C*********************************************************************
C* PICBUF STUFF
      PARAMETER NBUFSZ=300
      PARAMETER NCOLSMAX=256
      INTEGER IBUF(NBUFSZ),IBUF2(NBUFSZ)
      INTEGER IARRAY(NCOLSMAX),INAME(20),IOTNM(20)
      IMPLICIT INTEGER(A,B,C,R,X,Y)
C* I/O CONSTANTS
      PARAMETER IN=11
      PARAMETER OUT=10
C
C ACCEPT INPUT
C
      ACCEPT"NAME OF THE INPUT FILE:-->"
      READ(IN,40)INAME(1)
 40   FORMAT(S40)
      ACCEPT"NAME OF THE OUTPUT FILE:-->"
      READ(IN,41)IOTNM(1)
 41   FORMAT(S40)

C
C  PICBUFF FOR INPUT FILE
C
      CALL PICFMT(IBUF,INAME,IHDR)
      CALL GFMT(IBUF,NROWS,NCOLS,NBITS,IMODE)
      CALL MAKB(IBUF)
      CALL PICIN(IBUF,INAME,IHDR)

      CALL DFILW (IOTNM,IER) ;delete output file if it exists
      IF(IER.NE.1)GOTO 90
90    CONTINUE

C
C  PICBUFF FOR OUTPUT FILE
C
```

```
      CALL GFMT(IBUF,NROWS,NCOLS,NBITS,IMODE)
      CALL PFMT(IBUF2,NROWS,NCOLS,NBITS,IMODE)
      CALL MAKB(IBUF2)

C
C  GET POSITION
C
 4    TYPE" "
      TYPE"ROW,COLUMN OF THE UPPER LEFT CORNER"
      ACCEPT"OF THE ORIGINAL TEMPLATE-->",R,C
      XL=C
      YL=R

C
C  ASSUME X INCREASES TO THE RIGHT AND Y INCREASES DOWNWARD
C
      TYPE" "
      TYPE"ROW,COLUMN OF THE LOWER RIGHT CORNER"
      ACCEPT"OF THE ORIGINAL TEMPLATE-->",R,C
      XH=C
      YH=R

C
C  VERIFY PROPER ORIENTATION
C
      IF((XL.LT.XH).AND.(YL.LT.YH)) GO TO 5          ; OKAY
      TYPE" "
      TYPE"ORIENTATION IS INCORRECT...RE-ENTER--->"  ; NOT OKAY
      GO TO 4

C
C  CALCULATE MID PT.
C
 5    XM=XL+(XH-XL)/2
      YM=YL+(YH-YL)/2
      XM=INT(XM)                  ;INSURE INTEGERS
      YM=INT(YM)                  ;INSURE INTEGERS

C
C  MOVE UPPER LEFT CORNER
C
      AXL=XL
      AYL=YL
      AXH=XM-1
      AYH=YM-1

      BXH=256
      BXL=256-(AXH-AXL)
      BYL=256-(AYH-AYL)

      CALL MOVE(AXL,AXH,AYL,AYH,BXL,BXH,BYL,IBUF,IBUF2,IARRAY)
```

```
C
C  MOVE LOWER LEFT CORNER
C
      AXL=XL
      AXH=XM-1
      AYL=YM
      AYH=YH

      BYL=1
      BXH=256
      BXL=256-(BXH-BXL)

      CALL MOVE(AXL,AXH,AYL,AYH,BXL,BXH,BYL,IBUF,IBUF2,IARRAY)

C
C  MOVE UPPER RIGHT CORNER
C
      AXL=XM
      AYL=YL
      AXH=XH
      AYH=YM-1

      BXL=1
      BXH=1+(AXH-AXL)
      BYL=256-(AYH-AYL)

      CALL MOVE(AXL,AXH,AYL,AYH,BXL,BXH,BYL,IBUF,IBUF2,IARRAY)

C
C  MOVE LOWER RIGHT CORNER
C
      AXL=XM
      AYL=YM
      AXH=XH
      AYH=YH

      BXL=1
      BYL=1
      BXH=1+(AXH-AXL)

      CALL MOVE(AXL,AXH,AYL,AYH,BXL,BXH,BYL,IBUF,IBUF2,IARRAY)

      CALL PICOUT(IBUF2,IOTNM,IHDR)        ;write to file
      CALL RELB(IBUF)                      ;close buffers
      CALL RELB(IBUF2)

      CALL RESET
      STOP"<7><7><7><7>SPIN"
      END
```

```
C*******************************************************************
C                                                                  *
C MOVE.FR -DG FORTRAN 5- LT RICHARD L. MILLS    27 JULY 1984       *
C                                                                  *
C     CALLED BY:                                                   *
C         SPIN.FR                                                  *
C                                                                  *
C*******************************************************************
C   Subroutine MOVE will do the actual pixel duplication in        *
C   the output file (region B) from the input file (region A).     *
C*******************************************************************

      SUBROUTINE MOVE(AXL,AXH,AYL,AYH,BXL,BXH,BYL,IBUF,IBUF2,
     +                                                   IARRAY)

      IMPLICIT INTEGER (A,B,X,Y)
      TYPE"MOVING PIXELS"
      ILEN=AXH-AXL+1
      IROW2=BYL
      ICOL=AXL
      ICOL2=BXL
      DO 200 IROW=AYL,AYH
          CALL GROW(IBUF,IROW,ICOL,ILEN,IARRAY)
          CALL PROW(IBUF2,IROW2,ICOL2,ILEN,IARRAY)
          IROW2=IROW2+1
200   CONTINUE
      RETURN
      END
```

```
C     PROGRAM VDTOCP - DG FORTRAN 5 - CAPT ROBERT WELLS, DEC 1983

C     THIS PROGRAM CONVERTS A 256 X 256 PACKED VIDEO FILE TO A
C     256 X 256 COMPLEX FILE

      INTEGER IFLNM(7),OFLNM(7)
      INTEGER MAIN(7),F3(7),MS(2),S1(2),S2(2),S3(2)
      INTEGER TEMP(512),IN(2048)
      COMPLEX OUT(2048)

      CALL IOF(2,MAIN,IFLNM,OFLNM,F3,MS,S1,S2,S3)

      CALL OPEN(1,IFLNM,1,IER)
      CALL CHECK(IER)
      CALL OPEN(2,OFLNM,3,IER)
      CALL CHECK(IER)

      DO 3 I=0,31

         CALL RDBLK(1,2*I,TEMP,2,IER)
         CALL CHECK(IER)
         CALL UNPACK(512,TEMP,IN)

         DO 2 J=1,2048
            OUT(J)=CMPLX(FLOAT(IN(J)),0.0)
   2     CONTINUE

         CALL WRBLK(2,32*I,OUT,32,IER)
         CALL CHECK(IER)

   3 CONTINUE

     CALL RESET
     STOP VDTOCP
     END
```

```
C*********************************************************************
C                                                                    *
C   VCM.FR - DG FORTRAN 5 - LT RICHARD MILLS, AUG 1984               *
C                                                                    *
C     Originally CMULT - by Capt Robert Wells, Dec 1983              *
C     which performed complex multiplication of two 256 X            *
C     256 complex files on a point by point basis.                   *
C                                                                    *
C     VCM will do the same complex multiplication, but for           *
C     two complex files which are N x N.  N must be a power of       *
C     two and be 256 or less due to constraints of the 2DFFT         *
C     programs.                                                      *
C                                                                    *
C     RELOAD LINE:                                                   *
C       RLDR VCM @FLIB@                                              *
C                                                                    *
C     COMMAND LINE:                                                  *
C       VCM                                                          *
C     Program will ask for input and output file name, create        *
C       the output file and then ask for file size.                  *
C                                                                    *
C*********************************************************************

      COMPLEX IN1(1024),IN2(1024),OUT(21024
      INTEGER INFLNM1(7),INFLNM2(7),OFLNM1(7)
C
C ACCEPT INPUT
C
      ACCEPT"ENTER INPUT FILENAME #1 ->"
      READ(11,1)INFLNM1(1)
      ACCEPT"ENTER INPUT FILENAME #2 ->"
      READ(11,1)INFLNM2(1)
    1 FORMAT(S13)
      ACCEPT"ENTER OUTPUT FILENAME ->"
      READ(11,1)OFLNM1(1)

      CALL OPEN(1,INFLNM1,1,IER)
      CALL CHECK(IER)
      CALL OPEN(2,INFLNM2,1,IER)
      CALL CHECK(IER)

      CALL DFILW (OFLNM1,IER)
      CALL CFILW (OFLNM1,2,KER)
      CALL CHECK(KER)
      OPEN 3, OFLNM1,ATT="OR",ERR=100

100   CONTINUE
5     TYPE" "
      TYPE"ENTER SIZE OF FILES:"
      ACCEPT" 256, 128, 64, OR 32 -->",ISIZE
```

```
C
C  TEST FOR ONLY THE ALLOWABLE SIZES
C
      IF(ISIZE.EQ.256) GO TO 6
      IF(ISIZE.EQ.128) GO TO 6
      IF(ISIZE.EQ.64) GO TO 6
      IF(ISIZE.EQ.32) GO TO 6
      GO TO 5
6     CONTINUE    ; size is okay

      IFACTOR=256/ISIZE                  ;compute scaling factor
      ITIMES=64/IFACTOR**2-1

      DO 3 I=0,ITIMES

         CALL RDBLK(1,16*I,IN1,16,IER)
         CALL CHECK(IER)
         CALL RDBLK(2,16*I,IN2,16,IER)
         CALL CHECK(IER)

         DO 2 J=1,1024
            OUT(J)=IN1(J)*CONJG(IN2(J))
    2    CONTINUE

         CALL WRBLK(3,16*I,OUT,16,IER)
         CALL CHECK(IER)

    3 CONTINUE

      CALL RESET
      STOP"<7><7><7>VCM"
      END
```

```
C*****************************************************************
C                                                                *
C  FVCM.FR - DG FORTRAN 5 - LT RICHARD MILLS, 19 SEPT 1984      *
C                                   UPDATED 25 SEPT 1984        *
C                                   UPDATED  3 OCT  1984        *
C                                   UPDATED  4 OCT  1984        *
C                                                                *
C        FVCM (Filtered-Variable-Complex-Multiplication) is     *
C        similar to VCM which will do complex multiplication of *
C        any two complex files which are N x N, where N is a    *
C        power of two.  FVCM differs due to the fact that it    *
C        filters out everything above a given harmonic out of   *
C        the template file.  This program assumes that the      *
C        template file was centered at the origin before the    *
C        2DFFT was taken.                                        *
C                                                                *
C        Presently only 64 x 64 and 256 x 256.                  *
C                                                                *
C      RELOAD LINE:                                              *
C        RLDR FVCM @FLIB@                                        *
C                                                                *
C      COMMAND LINE:                                             *
C        FVCM                                                    *
C                                                                *
C*****************************************************************

      COMPLEX IN1(1024),IN2(1024),OUT(1024)
      INTEGER INFLNM1(7),INFLNM2(7),OFLNM1(7)
C
C ACCEPT INPUT
C
      ACCEPT"ENTER INPUT FILENAME #1 ->"
      READ(11,1)INFLNM1(1)
      ACCEPT"ENTER INPUT FILENAME #2 ->"
      READ(11,1)INFLNM2(1)
    1 FORMAT(S13)
      ACCEPT"ENTER OUTPUT FILENAME ->"
      READ(11,1)OFLNM1(1)

      CALL OPEN(1,INFLNM1,1,IER)
      CALL CHECK(IER)
      CALL OPEN(2,INFLNM2,1,IER)
      CALL CHECK(IER)

      CALL DFILW (OFLNM1,IER)
      CALL CFILW (OFLNM1,2,KER)
      CALL CHECK(KER)
      OPEN 3, OFLNM1,ATT="OR",ERR=100

100   CONTINUE
```

```
5     TYPE" "
      TYPE"ENTER SIZE OF FILES:"
      ACCEPT" 256, or 64 -->",ISIZE

C
C  TEST FOR ONLY THE ALLOWABLE SIZES
C
      IF(ISIZE.EQ.256) GO TO 7
      IF(ISIZE.EQ.64) GO TO 6
      GO TO 5

7     TYPE" "                               ;determine filter size for
                                            ; 256 x 256 file.
      TYPE"ENTER '1'-'32' FOR FILTER OF SIZE 7-255"
      TYPE"           (FILTER SIZE INCREMENTS BY 8)  "
      TYPE"FOR EXAMPLE: 1 FOR A 7x7 FILTER, 2 FOR A 15X15 FILTER,"
      ACCEPT"              3 FOR A 23X23 FILTER, ETC...  --> ",IFS

      IF(IFS.GT.32)GO TO 7
      IF(IFS.LT.1)GO TO 7

      IFSIZE=IFS*8-1
      TYPE"  "
      TYPE"FILTER TO BE USED IS ",IFSIZE," BY ",IFSIZE,"."
      TYPE"  "
      ACCEPT"ENTER '1' IF THIS IS CORRECT; '0' IF NOT.  --> ",IOK
      IF(IOK.NE.1)GO TO 7                    ;redo input if incorrect

6     CONTINUE   ; size is okay
      TYPE"------- PROGRAM RUNNING -------"
      IFACTOR=256/ISIZE                      ;compute scaling factor
      ITIMES=64/IFACTOR**2-1
C
C FILL OUTPUT FILE WITH ZEROS
C
      DO 3 I=0,ITIMES

         DO 2 J=1,1024
            OUT(J)=(0.0,0.0)               ;fill array OUT with zeros
    2    CONTINUE

         CALL WRBLK(3,16*I,OUT,16,IER)  ;write array to file
         CALL CHECK(IER)

    3 CONTINUE

      IF(ISIZE.EQ.64)GO TO 899              ;file is 64x64
C
C MULTIPLY AND WRITE TO FILE THE DC AND FIRST THREE HARMONICS
C
```

```
C**********************
C  256 X 256          *
C**********************
699    I=32-IFS                                ;do first block

          CALL RDBLK(1,16*I,IN1,16,IER)
          CALL CHECK(IER)
          CALL RDBLK(2,16*I,IN2,16,IER)
          CALL CHECK(IER)
          DO 18 J=1,1024
             OUT(J)=(0.0,0.0)     ;set output array to zeros
   18     CONTINUE

          IWIDE=129-((IFSIZE-1)/2)

          ISTART=1024-3*ISIZE+IWIDE
                                   ;start before center of second row
          IEND=ISTART+2*ISIZE
                                   ;end on fourth row
          DO 17 IGO=ISTART,IEND,ISIZE
                                   ;increment by ISIZE
            ISTOP=IGO+IFSIZE-1
                                   ;do IFSIZE points on each row

            DO 16 J=IGO,ISTOP
               OUT(J)=IN1(J)*CONJG(IN2(J)) ;multiply input arrays
   16       CONTINUE

   17     CONTINUE

          CALL WRBLK(3,16*I,OUT,16,IER)
          CALL CHECK(IER)
C
C COMPUTE REMAING BLOCKS
C
       IRB1=33-IFS
       IRB2=31+IFS

       DO 700 I=IRB1,IRB2                      ;do remaining blocks

          CALL RDBLK(1,16*I,IN1,16,IER)
          CALL CHECK(IER)
          CALL RDBLK(2,16*I,IN2,16,IER)
          CALL CHECK(IER)

          DO 28 J=1,1024
             OUT(J)=(0.0,0.0)                  ;set output array to zeros
   28     CONTINUE

          ISTART=IWIDE            ;start before center of first row
          IEND=ISTART+3*ISIZE     ;end on fourth row
          DO 27 IGO=ISTART,IEND,ISIZE

            ISTOP=IGO+IFSIZE-1        ;do IFSIZE points on each row

            DO 26 J=IGO,ISTOP
               OUT(J)=IN1(J)*CONJG(IN2(J)) ;multiply input arrays
   26       CONTINUE

   27     CONTINUE

          CALL WRBLK(3,16*I,OUT,16,IER)
          CALL CHECK(IER)
  700  CONTINUE

       GO TO 900                              ;skip to end

C****************************************
C  64 X 64 WITH A 15 X 15 FILTER        *
C****************************************

899    IMID3=(ITIMES+1)/2                     ;block past center
       IMID2=IMID3-1                          ;block before center

       TYPE"A STANDARD 15 X 15 FILTER IS USED ON THE 64 X 64 FILE."

       I=IMID2                                ;do IMID2 block

          CALL RDBLK(1,16*I,IN1,16,IER)
          CALL CHECK(IER)
          CALL RDBLK(2,16*I,IN2,16,IER)
          CALL CHECK(IER)

          DO 58 J=1,1024
             OUT(J)=(0.0,0.0)              ;set output array to zeros
   58     CONTINUE

          ISTART=1024-7*ISIZE+27
                                  ;start before center of second row
                                  ;27=33(center)-((15-1)/2)
          IEND=ISTART+6*ISIZE     ;end on sixteenth row

          DO 57 IGO=ISTART,IEND,ISIZE      ;increment by ISIZE

            ISTOP=IGO+14                   ;do 15 points on each row

            DO 56 J=IGO,ISTOP
               OUT(J)=IN1(J)*CONJG(IN2(J)) ;multiply input arrays
   56       CONTINUE
   57     CONTINUE

          CALL WRBLK(3,16*I,OUT,16,IER)
          CALL CHECK(IER)
```

```
C*****
      I=IMID3                                     ;do IMID3 block

         CALL RDBLK(1,16*I,IN1,16,IER)
         CALL CHECK(IER)
         CALL RDBLK(2,16*I,IN2,16,IER)
         CALL CHECK(IER)

         DO 78 J=1,1024
            OUT(J)=(0.0,0.0)                     ;set output array to zeros
   78    CONTINUE

         ISTART=27                   ;start before center of first row
         IEND=ISTART+7*ISIZE         ;end on seventh row
         DO 77 IGO=ISTART,IEND,ISIZE

           ISTOP=IGO+14                     ;do seven points on each row

           DO 76 J=IGO,ISTOP
              OUT(J)=IN1(J)*CONJG(IN2(J)) ;multiply input arrays
   76      CONTINUE

   77    CONTINUE

         CALL WRBLK(3,16*I,OUT,16,IER)
         CALL CHECK(IER)

900   CALL RESET
      STOP"<7><7><7>FVCM"
      END
```

```
C  PROGRAM CPTOVD - DG FORTRAN 5 - CAPT ROBERT WELLS, DEC 1983

C  THIS PROGRAM FINDS THE MAXIMUM AND MINIMUM OF A 256 X 256
C  COMPLEX FILE AND GIVES THE LOCATION OF THE MAXIMUM. ALSO, THE
C  PROGRAM CONVERTS THE FILE TO PACKED VIDEO FORMAT BY PERFORMING
C  A LINEAR SCALING TO THE RANGE OF 0 - 15 AND TRUNCATING.

      INTEGER IFLNM(7),OFLNM(7)
      INTEGER MAIN(7),F3(7),MS(2),S1(2),S2(2),S3(2)
      INTEGER TEMP(512),OUT(2048),IMAX,JMAX,MAXCOL,MAXROW
      COMPLEX IN(2048)
      REAL RMAG,MAXVAL,MINVAL,RANGE

      CALL IOF(2,MAIN,IFLNM,OFLNM,F3,MS,S1,S2,S3)

      CALL OPEN(1,IFLNM,3,IER)
      CALL CHECK(IER)
      CALL OPEN(2,OFLNM,3,IER)
      CALL CHECK(IER)

      MAXVAL=0.0
      MINVAL=1E50

      DO 3 I=0,31      ; FIND MAX AND MIN

         CALL RDBLK(1,32*I,IN,32,IER)
         CALL CHECK(IER)

         DO 3 J=1,2048

            RMAG=CABS(IN(J))

            IF(RMAG.LE.MAXVAL)GO TO 2
               MAXVAL=RMAG
               IMAX=I
               JMAX=J

 2          IF(RMAG.LT.MINVAL)MINVAL=RMAG

 3 CONTINUE

      RANGE=MAXVAL-MINVAL

      MAXCOL=MOD(JMAX-1,256)+1
      MAXROW=8*IMAX+INT((JMAX-1)/256)+1

      TYPE"MAX COL = ",MAXCOL    ; REPORT MAX INFO.
      TYPE"MAX ROW = ",MAXROW

      TYPE"MAX VAL = ",MAXVAL
      TYPE"MIN VAL = ",MINVAL
      DO 5 I=0,31

         CALL RDBLK(1,32*I,IN,32,IER)
         CALL CHECK(IER)

         DO 4 J=1,2048
            OUT(J)=ANINT(15.0*(CABS(IN(J))-MINVAL)/RANGE)
4        CONTINUE

         CALL REPACK(512,OUT,TEMP)
         CALL WRBLK(2,2*I,TEMP,2,IER)
         CALL CHECK(IER)

5     CONTINUE

      CALL RESET
      STOP"<7><7><7><7>CPTOVD"
      END
```

```
C**********************************************************************
C                                                                     *
C SMALL.FR   -DG FORTRAN 5-   LT. RICHARD L. MILLS   6 AUG 1984        *
C                                                                     *
C    This program will accept the name of a video file and            *
C    create a new, smaller, file which contains only specific         *
C    video info.  The only other info needed is the X-Y               *
C    coordinates (Row-Column) of the original position of             *
C    interest and the size of the area to be copied into the          *
C    smaller file.                                                    *
C                                                                     *
C    RELOAD LINE:                                                     *
C      RLDR SMALL F5PICBUF.LB @FLIB@                                  *
C                                                                     *
C    COMMAND LINE:                                                    *
C      SMALL                                                          *
C                                                                     *
C**********************************************************************
C* PICBUF STUFF
      PARAMETER NBUFSZ=300
      PARAMETER NCOLSMAX=256
      INTEGER IBUF(NBUFSZ),IBUF2(NBUFSZ)
      INTEGER IARRAY(NCOLSMAX),INAME(20),IOTNM(20)
      IMPLICIT INTEGER(A,B,C,R,X,Y)
C* I/O CONSTANTS
      PARAMETER IN=11
      PARAMETER OUT=10
C
C ACCEPT INPUTS
C
      ACCEPT"NAME OF THE INPUT FILE:-->"
      READ(IN,40)INAME(1)
 40   FORMAT(S40)
      ACCEPT"NAME OF THE OUTPUT FILE:-->"
      READ(IN,41)IOTNM(1)
 41   FORMAT(S40)
C
C  GET SIZE OF NEW IMAGE
C
5     TYPE" "
      TYPE"ENTER SIZE OF AREA OF INTEREST:"
      ACCEPT" 128, 64, 32: ",ISIZE
C
C  TEST FOR ONLY THE ALLOWABLE SIZES
C   (DUE TO CONSTRANTS ON FFT & IFFT PROGRAMS)
C
      IF(ISIZE.EQ.128) GO TO 6
      IF(ISIZE.EQ.64) GO TO 6
      IF(ISIZE.EQ.32) GO TO 6
      GO TO 5
6     CONTINUE   ; size is okay
```

```
      IFACTOR=256/ISIZE         ;calculate reduction factor
C
C  GET POSITION
C
      TYPE" "
      TYPE"ROW,COLUMN OF THE POINT OF INTERST "
      ACCEPT"IN THE ORIGINAL IMAGE-->",R,C
      X=C
      Y=R
C
C  PICBUFF FOR INPUT FILE
C
      CALL PICFMT(IBUF,INAME,IHDR)
      CALL GFMT(IBUF,NROWS,NCOLS,NBITS,IMODE)
      CALL MAKB(IBUF)
      CALL PICIN(IBUF,INAME,IHDR)

      CALL DFILW (IOTNM,IER)     ;delete output file if it exists
      IF(IER.NE.1)GOTO 90
90    CONTINUE
C
C  PICBUFF FOR OUTPUT FILE
C
      CALL GFMT(IBUF,NROWS,NCOLS,NBITS,IMODE)
      NROWS2=NROWS/IIACTOR
      NCOLS2=NCOLS/IFACTOR
      CALL PFMT(IBUF2,NROWS2,NCOLS2,NBITS,IMODE)
      CALL MAKB(IBUF2)
C
C  AINSURE PROPER DISTANCE FROM BORDERS
C
      IDIST=ISIZE/2
      IF(IDIST.LE.R)GO TO 20
      R=IDIST
      TYPE" "
      TYPE" TOO CLOSE TO IMAGE BORDER,"
      WRITE FREE(OUT)" NEW ROW IS ",R

20    IF(IDIST.LE.C)GO TO 21
      C=IDIST
      TYPE" "
      TYPE" TOO CLOSE TO IMAGE BORDER,"
      WRITE FREE(OUT)"NEW COLUMN IS ",C

21    IF((256-R).GE.IDIST)GO TO 22
      R=256-IDIST
      TYPE" "
      TYPE" TOO CLOSE TO IMAGE BORDER,"
      WRITE FREE(OUT)" NEW ROW IS ",R

22    IF((256-C).GE.IDIST)GO TO 23
      C=256-IDIST
```

```
        TYPE" "
        TYPE" TOO CLOSE TO IMAGE BORDER,"
        WRITE FREE(OUT)"NEW COLUMN IS ",C
23      CONTINUE            ;R,C are at least IDIST from image border
        TYPE" "
        TYPE"MOVING PIXELS"
        ISTART=R-IDIST
        ISTOP=ISTART+ISIZE-1
        IROW2=1
        ICOL=C-IDIST
        ICOL2=1
        DO 200 IROW=ISTART,ISTOP
            CALL GROW(IBUF,IROW,ICOL,ISIZE,IARRAY)
            CALL PROW(IBUF2,IROW2,ICOL2,ISIZE,IARRAY)
            IROW2=IROW2+1
200     CONTINUE

        CALL PICOUT(IBUF2,IOTNM,IHDR)         ;write to file
        CALL RELB(IBUF)                       ;close buffers
        CALL RELB(IBUF2)

        CALL RESET
        STOP"<7><7><7><7>SMALL"
        END
```

```
C*****************************************************************
C                                                                *
C VSPIN.FR   -DG FORTRAN 5-    LT RICHARD L. MILLS  17 AUG 1984  *
C                                      UPDATED    7 SEPT 1984    *
C                                                                *
C    This program will accept the name of a file which contains  *
C    a template and create a new file which places the template  *
C    center at the origin of the video file.  This is much like  *
C    the program SPIN.FR except VSPIN.FR (variable-Spin)works    *
C    on variable sized video files and creates files of the      *
C    same size.  The input is assumed to be centered, there-     *
C    fore no coordinates are needed.                             *
C                                                                *
C    RELOAD LINE:                                                *
C      RLDR VSPIN F5PICBUF.LB @FLIB@                             *
C                                                                *
C    COMMAND LINE:                                               *
C      VSPIN                                                     *
C                                                                *
C*****************************************************************
C* PICBUF STUFF
      PARAMETER NBUFSZ=300
      PARAMETER NCOLSMAX=256
      INTEGER IBUF(NBUFSZ),IBUF2(NBUFSZ)
      INTEGER IARRAY(NCOLSMAX),INAME(20),IOTNM(20)
      IMPLICIT INTEGER(A,B,C,R,X,Y)
C* I/O CONSTANTS
      PARAMETER IN=11
      PARAMETER OUT=10
C
C ACCEPT INPUT
C
      ACCEPT"NAME OF THE INPUT FILE:-->"
      READ(IN,40)INAME(1)
 40   FORMAT(S40)
      ACCEPT"NAME OF THE OUTPUT FILE:-->"
      READ(IN,41)IOTNM(1)
 41   FORMAT(S40)

5     TYPE" "
      TYPE"ENTER SIZE OF INPUT FILE:"
      ACCEPT" 256, 128, 64, OR 32 -->",ISIZE
C
C TEST FOR ONLY THE ALLOWABLE SIZES
C
      IF(ISIZE.EQ.256) GO TO 6
      IF(ISIZE.EQ.128) GO TO 6
      IF(ISIZE.EQ.64) GO TO 6
      IF(ISIZE.EQ.32) GO TO 6
      GO TO 5
6     CONTINUE   ; size is okay
```

```
C
C  PICBUFF FOR INPUT FILE
C
      NROWS=ISIZE
      NCOLS=ISIZE
      NBITS=4
      IMODE=1
      IHDR=1
      CALL PFMT(IBUF,NROWS,NCOLS,NBITS,IMODE)
      CALL MAKB(IBUF)
      CALL PICIN(IBUF,INAME,IHDR)

      CALL DFILW (IOTNM,IER)     ;delete output file if it exists
      IF(IER.NE.1)GOTO 90
90    CONTINUE
C
C  PICBUFF FOR OUTPUT FILE
C
      CALL GFMT(IBUF,NROWS,NCOLS,NBITS,IMODE)
      CALL PFMT(IBUF2,NROWS,NCOLS,NBITS,IMODE)
      CALL MAKB(IBUF2)
C
C  CALCULATE MID PT., QUARTERED SIZE, CENTER STARTING PT.
C
      XM=ISIZE/2
      YM=ISIZE/2
      ILEN=ISIZE/2
      ISTART=(ISIZE/2)+1
C
C  MOVE UPPER LEFT CORNER
C
      TYPE"MOVING PIXELS"
      IROW2=YM+1
      ICOL=1
      ICOL2=XM+1
      DO 200 IROW=1,ILEN
          CALL GROW(IBUF,IROW,ICOL,ILEN,IARRAY)
          CALL PROW(IBUF2,IROW2,ICOL2,ILEN,IARRAY)
          IROW2=IROW2+1
200   CONTINUE
C
C  MOVE UPPER RIGHT CORNER
C
      TYPE"MOVING PIXELS"
      IROW2=YM+1
      ICOL=XM+1
      ICOL2=1
      DO 400 IROW=1,ILEN
          CALL GROW(IBUF,IROW,ICOL,ILEN,IARRAY)
          CALL PROW(IBUF2,IROW2,ICOL2,ILEN,IARRAY)
          IROW2=IROW2+1
400   CONTINUE
```

```
C
C  MOVE LOWER LEFT CORNER
C
      TYPE"MOVING PIXELS"
      IROW2=1
      ICOL=1
      ICOL2=XM+1
      DO 300 IROW=ISTART,ISIZE
          CALL GROW(IBUF,IROW,ICOL,ILEN,IARRAY)
          CALL PROW(IBUF2,IROW2,ICOL2,ILEN,IARRAY)
          IROW2=IROW2+1
300   CONTINUE
C
C  MOVE LOWER RIGHT CORNER
C
      TYPE"MOVING PIXELS"
      IROW2=1
      ICOL=YM+1
      ICOL2=1
      DO 500 IROW=ISTART,ISIZE
          CALL GROW(IBUF,IROW,ICOL,ILEN,IARRAY)
          CALL PROW(IBUF2,IROW2,ICOL2,ILEN,IARRAY)
          IROW2=IROW2+1
500   CONTINUE

      CALL PICOUT(IBUF2,IOTNM,1)           ;write to file
      CALL RELB(IBUF)                      ;close buffers
      CALL RELB(IBUF2)

      CALL RESET
      STOP"<7><7><7><7>VSPIN"
      END
```

```
C*****************************************************************
C                                                                *
C     NORM.FR - DG FORTRAN 5 - LT RICHARD MILLS, 11 SEPT 1984    *
C                                                                *
C       This program will accept a template file and a target   *
C       file and create a third file which will be a new target *
C       file with its energy normalized to that of the template.*
C       File sizes may vary; 32, 64, 128, or 256.  (Normalized  *
C       energy may not exactly equal that of the template due   *
C       to round off errors.)                                   *
C       This program is similar to NORM2.FR except for the fact *
C       NORM.FR will automatically continue the normalization   *
C       process repetitively until it can no longer do so due   *
C       to the round off error.                                 *
C                                                                *
C     RELOAD LINE:                                               *
C       RLDR NORM UNPACK REPACK @FLIB@                           *
C                                                                *
C     COMMAND LINE:                                              *
C       NORM                                                     *
C                                                                *
C*****************************************************************

      INTEGER TEMP(256),IN(1024)
      INTEGER INFLNM(7),OFLNM1(7),OFLNM2(7)
      REAL OUT(1024)
      DOUBLE PRECISION ENERGY1,ENERGY2,ENERGY3
C
C ACCEPT INPUT
C
      ACCEPT"ENTER TEMPLATE FILENAME ->"
      READ(11,1)INFLNM(1)
    1 FORMAT(S13)
      ACCEPT"ENTER TARGET FILENAME ->"
      READ(11,1)OFLNM1(1)
      ACCEPT"ENTER NEW NAME OF NORMALIZED TARGET ->"
      READ(11,1)OFLNM2(1)

      CALL OPEN(1,INFLNM,1,IER)
      CALL CHECK(IER)

      CALL OPEN(2,OFLNM1,1,IER)
      CALL CHECK(IER)

      CALL DFILW (OFLNM2,IER)
      CALL CFILW (OFLNM2,2,KER)
      CALL CHECK(KER)
      OPEN 3, OFLNM2,ATT="OR",ERR=100
100   CONTINUE
5     TYPE" "
      TYPE"ENTER SIZE OF FILES:"
      ACCEPT" 256, 128, 64, OR 32 -->",ISIZE
```

```
C
C  TEST FOR ONLY THE ALLOWABLE SIZES
C
      IF(ISIZE.EQ.256) GO TO 6
      IF(ISIZE.EQ.128) GO TO 6
      IF(ISIZE.EQ.64) GO TO 6
      IF(ISIZE.EQ.32) GO TO 6
      GO TO 5
6     CONTINUE   ; size is okay

      IFACTOR=256/ISIZE          ;calculate reduction factor
      ITIMES=64/IFACTOR**2-1

      ENERGY1=0                  ;initialize energies to zero
      ENERGY2=0
C
C READ TEMPLATE ENERGY
C
      DO 3 I=0,ITIMES

         CALL RDBLK(1,I,TEMP,1,IER)
         CALL CHECK(IER)
         CALL UNPACK(256,TEMP,IN)

         DO 2 J=1,1024
            ENERGY1=ENERGY1+(IN(J))**2
    2    CONTINUE

    3 CONTINUE
C
C READ TARGET ENERGY
C
      DO 13 I=0,ITIMES

         CALL RDBLK(2,I,TEMP,1,IER)
         CALL CHECK(IER)
         CALL UNPACK(256,TEMP,IN)

         DO 12 J=1,1024
            ENERGY2=ENERGY2+(IN(J))**2
   12    CONTINUE

   13 CONTINUE
      ENERGY3=ENERGY2
C
C DISPLAY DATA
C
      TYPE" "
      TYPE"TEMPLATE ENERGY :          ",ENERGY1
      TYPE"ORIGINAL TARGET ENERGY :  ",ENERGY2
50    RATIO=DSQRT(ENERGY1/ENERGY2)
      TYPE"SQ. RT. ENERGY RATIO :    ",RATIO
```

```
C
C READ TARGET ENERGY AND NORMALIZE
C
      ENERGY2=0
      DO 23 I=0,ITIMES

         CALL RDBLK(2,I,TEMP,1,IER)
         CALL CHECK(IER)
         CALL UNPACK(256,TEMP,IN)
         DO 22 J=1,1024
            OUT(J)=IN(J)*RATIO
            IN(J)=IFIX(OUT(J)+0.5)    ;round off
            IF(IN(J).GT.15) IN(J)=15
            ENERGY2=ENERGY2+(IN(J))**2
   22    CONTINUE
      CALL REPACK(256,IN,TEMP)
      CALL WRBLK(3,I,TEMP,1,IER)
      CALL CHECK(IER)

   23 CONTINUE
      TYPE"NEW TARGET ENERGY :       ",ENERGY2
C
C TEST IF ENERGY IS AS CLOSE AS IT CAN GET
C
      IF(ENERGY3.EQ.ENERGY2)GO TO 51
      ENERGY3=ENERGY2

      CALL CLOSE(2,IER)
      CALL CHECK(IER)
      CALL DFILW ("HOLD.VD",IER)
      CALL CFILW ("HOLD.VD",2,KER)
      CALL CHECK(KER)
      OPEN 4, "HOLD.VD",ATT="OR",ERR=101
101   CONTINUE
C
C  TRANSFER OUTPUT TO HOLD.VD AND CHANGE CHANNELS
C
      DO 33 I=0,ITIMES

         CALL RDBLK(3,I,TEMP,1,IER)
         CALL CHECK(IER)
         CALL WRBLK(4,I,TEMP,1,IER)
         CALL CHECK(IER)
33    CONTINUE
      CALL CLOSE(4,IER)
      CALL CHECK(IER)
      CALL OPEN(2,"HOLD.VD",1,IER)
      CALL CHECK(IER)
      GO TO 50
```

```
51      CALL RESET
        CALL DFILW("HOLD.VD",IER)
        CALL CHECK(IER)
        STOP"<7><7><7><7>NORM"
        END
```

```
C*****************************************************************
C                                                                *
C  VTC.FR  - DG FORTRAN 5 - LT RICHARD MILLS, AUG 1984           *
C                                                                *
C     Originally VDTOCP - by Capt Robert Wells, Dec 1983         *
C     which converted a 256 X 256 packed vedio file to a         *
C     256 X 256 complex file.                                    *
C                                                                *
C     VTC will do the same complex file construction but for     *
C     any packed video file which is N x N.  N must be a power   *
C     of two and be 256 or less due to constrants on the 2DFFT   *
C     programs.                                                  *
C                                                                *
C     RELOAD LINE:                                               *
C       RLDR VTC UNPACK @FLIB@                                   *
C                                                                *
C     COMMAND LINE:                                              *
C       VTC                                                      *
C     Program will ask for input and output file name, create    *
C     the output file and then ask for file size.                *
C                                                                *
C*****************************************************************

      INTEGER TEMP(256),IN(1024)
      INTEGER INFLNM(7),OFLNM1(7)
      COMPLEX OUT(2048)
C
C GET INPUTS
C
      ACCEPT"ENTER INPUT FILENAME ->"
      READ(11,1)INFLNM(1)
    1 FORMAT(S13)
      ACCEPT"ENTER OUTPUT FILENAME ->"
      READ(11,1)OFLNM1(1)

      CALL OPEN(1,INFLNM,1,IER)
      CALL CHECK(IER)

      CALL DFILW (OFLNM1,IER)
      CALL CFILW (OFLNM1,2,KER)
      CALL CHECK(KER)
      OPEN 2, OFLNM1,ATT="OR",ERR=100
100   CONTINUE
C
C GET FILE SIZE
C
5     TYPE" "
      TYPE"ENTER SIZE OF INPUT FILE:"
      ACCEPT" 256, 128, 64, OR 32 -->",ISIZE
C
C  TEST FOR ONLY THE ALLOWABLE SIZES
C
```

```
      IF(ISIZE.EQ.256) GO TO 6
      IF(ISIZE.EQ.128) GO TO 6
      IF(ISIZE.EQ.64) GO TO 6
      IF(ISIZE.EQ.32) GO TO 6
      GO TO 5
6     CONTINUE   ; size is okay

      IFACTOR=256/ISIZE                 ;get scaling factor
      ITIMES=64/IFACTOR**2-1
C
C DO CONVERSION TO COMPLEX
C
      DO 3 I=0,ITIMES

         CALL RDBLK(1,I,TEMP,1,IER)
         CALL CHECK(IER)
         CALL UNPACK(256,TEMP,IN)

         DO 2 J=1,1024
            OUT(J)=CMPLX(FLOAT(IN(J)),0.0)
                                       ;set imaginary part to zero
    2    CONTINUE

         CALL WRBLK(2,16*I,OUT,16,IER)
         CALL CHECK(IER)

    3 CONTINUE

      CALL RESET
      STOP"<7><7><7><7>VTC"
      END
```

```
C*****************************************************************
C                                                                *
C  CTV.FR - DG FORTRAN 5 - LT RICHARD MILLS, AUG 1984            *
C                                                                *
C      Originally CPTOVD - by Capt Robert Wells, Dec 1983        *
C      which converted a 256 X 256 complex file to a 256 X 256   *
C      packed video file.                                        *
C                                                                *
C      CTV will do the same complex file construction but for    *
C      and packed video file which is N x N.  N must be a power  *
C      of two and be 256 or less due to constrants on the 2DFFT  *
C      programs.                                                 *
C                                                                *
C      RELOAD LINE:                                              *
C        RLDR CTV IOF REPACK @FLIB@                              *
C                                                                *
C      COMMAND LINE:                                             *
C        CTV                                                     *
C      Program will ask for input and output file name, create   *
C        the output file and then ask for file size.             *
C                                                                *
C*****************************************************************

      INTEGER IFLNM(7),OFLNM(7)
      INTEGER TEMP(256),OUT(1024),IMAX,JMAX,MAXCOL,MAXROW
      COMPLEX IN(1024)
      REAL RMAG,MAXVAL,MINVAL,RANGE
C
C ACCEPT INPUT
C
      ACCEPT"ENTER INPUT FILENAME ->"
      READ(11,1)IFLNM(1)
    1 FORMAT(S13)
      ACCEPT"ENTER OUTPUT FILENAME ->"
      READ(11,1)OFLNM(1)

      CALL OPEN(1,IFLNM,1,IER)
      CALL CHECK(IER)

      CALL DFILW (OFLNM,IER)
      CALL CFILW (OFLNM,2,KER)
      CALL CHECK(KER)
      OPEN 2, OFLNM,ATT="OR",ERR=100

100   CONTINUE
5     TYPE" "
      TYPE"ENTER SIZE OF INPUT FILE:"
      ACCEPT" 256, 128, 64, OR 32 -->",ISIZE
C
C  TEST FOR ONLY THE ALLOWABLE SIZES
C
```

```
      IF(ISIZE.EQ.256) GO TO 6
      IF(ISIZE.EQ.128) GO TO 6
      IF(ISIZE.EQ.64) GO TO 6
      IF(ISIZE.EQ.32) GO TO 6
      GO TO 5
6     CONTINUE   ; size is okay

      IFACTOR=256/ISIZE                    ;compute scaling factor
      ITIMES=64/IFACTOR**2-1
C
      MAXVAL=0.0                           ;set initial values
      MINVAL=1E50

      DO 3 I=0,ITIMES                      ;find max and min

         CALL RDBLK(1,16*I,IN,16,IER)
         CALL CHECK(IER)

         DO 3 J=1,1024

            RMAG=CABS(IN(J))
            IF(RMAG.LE.MAXVAL)GO TO 2
               MAXVAL=RMAG
               IMAX=I
               JMAX=J

    2       IF(RMAG.LT.MINVAL)MINVAL=RMAG

    3 CONTINUE

      RANGE=MAXVAL-MINVAL                  ;find range of values

      MAXCOL=MOD(JMAX-1,256)+1
      MAXROW=8*IMAX+INT((JMAX-1)/256)+1

      TYPE"MAX COL = ",MAXCOL   ; REPORT MAX INFO.
      TYPE"MAX ROW = ",MAXROW

      TYPE"MAX VAL = ",MAXVAL
      TYPE"MIN VAL = ",MINVAL

      DO 8 I=0,ITIMES                      ;rewrite as video

         CALL RDBLK(1,16*I,IN,16,IER)
         CALL CHECK(IER)

         DO 4 J=1,1024
            OUT(J)=ANINT(15.0*(CABS(IN(J))-MINVAL)/RANGE)
    4    CONTINUE
```

```
      CALL REPACK(256,OUT,TEMP)
      CALL WRBLK(2,I,TEMP,1,IER)
      CALL CHECK(IER)

8 CONTINUE

  CALL RESET
  STOP"<7><7><7><7>CTV"
  END
```

```
C***********************************************************************
C                                                                      *
C     EVAL.FR - DG FORTRAN 5 - LT RICHARD MILLS, 12 SEPT 1984          *
C                                           Updated 17 SEPT 1984       *
C                                                                      *
C       This program will evaluate the correlation peaks after         *
C       normalization.  The evaluation is based on:                    *
C                                                                      *
C               1)  Peak location                                      *
C               2)  Peak sharpness/uniqueness.                         *
C                                                                      *
C       A hard copy output may be obtained by printing the             *
C       file "LPT".                                                    *
C                                                                      *
C                                                                      *
C     RELOAD LINE:                                                     *
C       RLDR EVAL F5PICBUF.LB @FLIB@                                   *
C                                                                      *
C     COMMAND LINE:                                                    *
C       EVAL                                                           *
C                                                                      *
C***********************************************************************
C  PICBUF STUFF
      PARAMETER NBUFSZ=300
      PARAMETER NCOLSMAX=512
      INTEGER IBUFF3(NBUFSZ)
      INTEGER IARRAY(NCOLSMAX)
C*
      PARAMETER NLIMIT=20
      PARAMETER PERCENT=5.0
      INTEGER TEMP(256),IN(1024)
      INTEGER MAINFL(7)
      INTEGER INFLNM(400)
      INTEGER ISIZE(NLIMIT)
      REAL DIST(NLIMIT)
      REAL MIDPT
      INTEGER MAXROW(NLIMIT),MAXCOL(NLIMIT)
      REAL SCORE1(NLIMIT),SCORE2(NLIMIT),IRANK(NLIMIT)
      REAL HIS(NLIMIT,16)

      CALL DFILW ("LPT",IER)           ;open up output file
      OPEN 12,"LPT"
C
C SET "HIS" ARRAY TO ZEROS
C
      DO 40 I=1,16
        DO 41 ICOUNT=1,NLIMIT
          HIS(ICOUNT,I)=0.0
 41     CONTINUE
 40   CONTINUE
```

```
        ICOUNT=0                                ;set counter
C
C ACCEPT INPUTS
C
        TYPE" "
        ACCEPT"ENTER FILENAME OF THE MAIN FILE ->"
        READ(11,1)MAINFL(1)

7       ICOUNT=ICOUNT+1                         ;increment counter
        TYPE" "
        ACCEPT"ENTER FILENAME ->"
        READ(11,1)INFLNM(ICOUNT*10)
      1 FORMAT(S20)

5       CONTINUE
        ACCEPT"ENTER SIZE OF FILE: 256, 128, 64, OR 32 -->",
     +                                          ISIZE(ICOUNT)
C
C  TEST FOR ONLY THE ALLOWABLE SIZES DUE TO FFT PROGRAMS
C
        IF(ISIZE(ICOUNT).EQ.256) GO TO 6
        IF(ISIZE(ICOUNT).EQ.128) GO TO 6
        IF(ISIZE(ICOUNT).EQ.64) GO TO 6
        IF(ISIZE(ICOUNT).EQ.32) GO TO 6
        GO TO 5
6       CONTINUE   ; size is okay

        IF(ICOUNT.EQ.NLIMIT)GO TO 8
        TYPE ICOUNT,"   FLIES SO FAR"
        ACCEPT "ENTER 1 FOR ANOTHER; 0 TO EVALUATE  ",IMORE
        IF(IMORE.EQ.1)GO TO 7
        TYPE" "
        TYPE"EVALUATING"
        TYPE" "
8       NUMFILES=ICOUNT
C
C INPUT COMPLETE, NOW EVALUATION

        DO 9 ICOUNT=1,NUMFILES
          CALL OPEN(1,INFLNM(ICOUNT*10),1,IER)
          CALL CHECK(IER)
C
C  CALCULATE MID PT.
C
          MIDPT=ISIZE(ICOUNT)/2+0.5
C
C PICBUFF FOR INPUT FILE
C
          NROWS=ISIZE
          NCOLS=ISIZE
          NBITS=4
          IMODE=1
      IHDR=1
      CALL PFMT(IBUF,NROWS,NCOLS,NBITS,IMODE)
      CALL MAKB(IBUF)
      CALL PICIN(IBUF,INFLNM(ICOUNT*10),IHDR)
C
C FIND MAX LOCATION
C
      IFLAG=0
      MAXROW(ICOUNT)=999      ;initailly set to very high values
      MAXCOL(ICOUNT)=999
      DO 20 IROW=1,ISIZE(ICOUNT)
        DO 21 ICOL=1,ISIZE(ICOUNT)
          CALL GPNT(IBUF,IROW,ICOL,IVAL)
          HIS(ICOUNT,IVAL+1)=HIS(ICOUNT,IVAL+1)+1.0
                          ;construct histrogram data array
          IF(IFLAG.EQ.1)GO TO 21         ;skip if found
            IF(IVAL.EQ.15)IFLAG=1        ;find max point
            IF(IFLAG.EQ.1)MAXROW(ICOUNT)=IROW
            IF(IFLAG.EQ.1)MAXCOL(ICOUNT)=ICOL
  21    CONTINUE
  20  CONTINUE
      CALL RELB(IBUF)
C
C EVALUATE MAX LOCATION
C
      DIST(ICOUNT)=SQRT((MIDPT-MAXCOL(ICOUNT))**2+
     +                  (MIDPT-MAXROW(ICOUNT))**2)
      SCORE1(ICOUNT)=1.0-(2*DIST(ICOUNT)/ISIZE(ICOUNT))
      IF(SCORE1(ICOUNT).LT.0.0)SCORE1(ICOUNT)=0.0
C
C DETERMINE SHARPNESS/UNIQUENESS OF PEAK
C
      PERCENTAREA=(PERCENT/100.0)*ISIZE(ICOUNT)**2
      ISUM1=0
      ISUM2=0
      J=16
 25   ISUM1=ISUM1+HIS(ICOUNT,J)
      IF(ISUM1.GT.PERCENTAREA)GO TO 26 ;"peakness" determined
      ISUM2=ISUM1
      J=J-1
      GO TO 25
 26   FRACTION=ISUM2/PERCENTAREA
      SPOT=J+FRACTION      ;determine score2 function's input
C
C EVALUATE THE SHARPNESS/UNIQUENESS
C
      SCORE2(ICOUNT)=SQRT((16.0-SPOT)/10.0)
      IF(SCORE2(ICOUNT).GT.1.0)SCORE2(ICOUNT)=1.0

      SCORE(ICOUNT)=SCORE1(ICOUNT)*SCORE2(ICOUNT)
```

```
          IRANK(ICOUNT)=IFIX(SCORE(ICOUNT)*10)+1
          IF(IRANK(ICOUNT).GT.10)IRANK(ICOUNT)=10

          CALL CLOSE(1,IER)
          CALL CHECK(IER)

9       CONTINUE        ;END OF EVALUATION LOOP

        WRITE (10,14)MAINFL(1)
        WRITE FREE (10) "FILE NAME              SCORE1      SCORE2
       +  TOTAL SCORE       RANK"
        WRITE FREE (10) "---------              ------      ------
       +  -----------       ----"
        DO 46 ICOUNT=1,NUMFILES
         ICNT=ICOUNT                  ;to save room
         ITEN=ICOUNT*10
         WRITE(10,2)INFLNM(ITEN),SCORE1(ICNT),SCORE2(ICNT),
        +SCORE(ICNT),IRANK(ICNT)
    2    FORMAT(1X,S13,5X,F8.6,5X,F8.6,7X,F8.6,9X,I2)
         IF(SCORE2(ICOUNT).GT.0.6.AND.SCORE1(ICOUNT).LT.0.7)
        +WRITE FREE (10) " POSSIBLE OFF-CENTERED PARTIAL TARGET"
 46     CONTINUE
C
C WRITE TO PRINTER, VIA "LPT"
C
        WRITE (12,14)MAINFL(1)
        WRITE (12,11)
        WRITE (12,12)
  14    FORMAT(1X,///////,30X,"EVALUATION RESULTS",/,25X,"SCENE
       +FILENAME: ",S16)
  11    FORMAT(8X," FILE NAME          SCORE1      SCORE2     TOTAL
       +SCORE     RANK")
  12    FORMAT(8X," ---------          ------      ------     ------
       +-----     ----")
        DO 47 ICOUNT=1,NUMFILES
         ICNT=ICOUNT                  ;to save room
         ITEN=ICOUNT*10
         WRITE(12,3)INFLNM(ITEN),SCORE1(ICNT),SCORE2(ICNT),
        +SCORE(ICNT),IRANK(ICNT)
         IF(SCORE2(ICOUNT).GT.0.6.AND.SCORE1(ICOUNT).LT.0.7)
        +WRITE (12,13)
  13     FORMAT(1H ,"       POSSIBLE OFF-CENTERED PARTIAL TARGET")
   3     FORMAT(9X,S13,3X,F8.6,4X,F8.6,5X,F8.6,8X,I2)
 47     CONTINUE

51      CALL RESET
        STOP"<7><7><7><7>EVAL   PRINT LPT  WILL GIVE HARD COPY"
        END
```

APPENDIX D

Support Software

This Appendix contains the support software used thoughout this thesis. Included are the following programs and subroutines:

VID.FR

WINDOW.FR

BI.FR

HISTO.FR

NEG.FR

UNPACK.FR

REPACK.FR

```
C*****************************************************************
C                                                                *
C VID.FR  - DG FORTRAN 4 - BY LT. RICHARD L. MILLS  10 AUG 1984 *
C                                                                *
C    VID will use the OCTEK to display files smaller than        *
C     256 x 256.                                                 *
C                                                                *
C    RELOAD COMMAND (from DP0:NMILLS on the NOVA)                *
C     RLDR MILLS:VID MILLS:WINDOW IACF4.LB NF4PICBUF.LB FORT.LB *
C                                                                *
C    EXECUTION COMMAND:                                          *
C     VID (from NOVA only)                                       *
C                                                                *
C*****************************************************************
C* PICBUF STUFF
        PARAMETER NBUFSZ=300
        PARAMETER NCOLSMAX=512
        INTEGER IBUFF3(NBUFSZ)
        INTEGER IARRAY(NCOLSMAX),INAME(20)
        INTEGER ICOL,IROW,IXP,IYP
        INTEGER DISP(200),ICT(120)
C
C INITIALIZE OCKTEK DISPLAY by Lt.J.Tilley
C
        IFIELD=0
        IX=0
        IXL=320
        IY=0
        IYL=240
        IENWV=15
        IENWC=15
        ICMAX=11
        CANGLE=0
        GRID=0
        CALL SINTRO (ICT,63K,IER)
        IF (IER.EQ.1) GOTO 51
                TYPE "INTRO ERROR CODE: ",IER
                STOP "UNABLE TO INITIALIZE"

51      CALL OPEN(3,"IACMON.XB",2,IER)
        IF (IER.NE.1) TYPE "WARNING: UNABLE TO ACCESS IACMON.XB"
        IF (IER.EQ.1) CALL LXB (ICT,3)

        CALL MPRUN (ICT,0,1)
        CALL BOXCUR(ICT,IXL,IYL)
        CALL HCTAB(ICT,IX,IY)
        CALL GREYSCALE (ICT,1)

        CALL PXFILL(ICT,15,1,320,1,240)
C
C End of Tilley's inits
C
```

```
C
C ACCEPT INPUT
C
100    ACCEPT"NAME OF THE INPUT FILE:-->"
       READ(11,40)INAME(1)
 40    FORMAT(S40)
5      ACCEPT"SIZE:-->",ISIZE
C
C  TEST FOR ONLY THE ALLOWABLE SIZES
C   (due to constrants on DIRECT & INVERSE)
C
       IF(I IZE.EQ.128) GO TO 6
       IF(ISIZE.EQ.64) GO TO 6
       IF(ISIZE.EQ.32) GO TO 6
       IF(ISIZE.EQ.16) GO TO 6
       IF(ISIZE.EQ.8) GO TO 6
       GO TO 5
6      CONTINUE    ; size is okay
C
C  GET POSITION
C
       TYPE" "
       ACCEPT"POSITION TO BE DISPLAYED (ROW,COLUMN)-->",IYP,IXP
       IXP=IXP+32

       ICOL=ISIZE
       IROW=ISIZE
C
C  PICBUFF FOR INPUT FILE
C
       NROWS=ISIZE
       NCOLS=ISIZE
       NBITS=4
       IMODE=1
       CALL PFMT(IBUFF3,NROWS,NCOLS,NBITS,IMODE)
       CALL MAKB(IBUFF3)
       CALL PICIN(IBUFF3,INAME,IHDR)

       CALL WINDOW(IBUFF3,ICOL,IROW,IXP,IYP,ICT)

       CALL RELB(IBUFF3)
       ITOGGLE=0
       TYPE" "
       ACCEPT"ENTER 1 TO INCLUDE ANOTHER PICTURE: ",ITOGGLE
       IF(ITOGGLE.EQ.1)GO TO 100

       CALL RESET
       STOP"<7><7><7>VID"
       END
```

```
C*****************************************************************
C                                                                *
C  WINDOW.FR -DG FORTRAN 4-     LT. J. TILLEY                    *
C            UPDATED BY LT. RICHARD L. MILLS  10 AUG 1984        *
C                                                                *
C      CALED BY:                                                 *
C        VID.FR                                                  *
C                                                                *
C*****************************************************************
C      The subroutine WINDOW.FR will do the actual writing of    *
C      the video blocks to the Octek board.                      *
C*****************************************************************

       SUBROUTINE WINDOW(IBUFF3,ICOL,IROW,IXP,IYP,ICT)

       INTEGER IBUFF3(300),ICOL,IROW,IXP,IYP    ;common arrays
       INTEGER DISP(200),ICT(120)

       DO 200 J=1,IROW

         CALL GROW(IBUFF3,J,1,ICOL,DISP)
         IXPOS=IXP
         IYPOS=IYP+J
         CALL WVBLK(ICT,DISP,IXPOS,ICOL,IYPOS,1)

200    CONTINUE

       RETURN
       END
```

```
C*********************************************************************
C                                                                    *
C     BI.FR  - DG FORTRAN 5 -  LT RICHARD MILLS, 12 SEPT 1984       *
C                                                                    *
C       This program will rewrite a video file into a binary        *
C       format so that it may be displayed with the program         *
C       PLOT3D.                                                      *
C                                                                    *
C     RELOAD LINE:                                                   *
C       RLDR BI UNPACK  @FLIB@                                       *
C                                                                    *
C     COMMAND LINE:                                                  *
C       BI                                                           *
C                                                                    *
C*********************************************************************

      INTEGER TEMP(256),IN(1024)
      INTEGER INFLNM(7),OFLNM2(7)
C
C ACCEPT INPUTS
C
      ACCEPT"ENTER INPUT FILENAME ->"
      READ(11,1)INFLNM(1)
    1 FORMAT(S13)
      ACCEPT"ENTER OUTPUT FILENAME ->"
      READ(11,1)OFLNM2(1)

      CALL OPEN(1,INFLNM,1,IER)
      CALL CHECK(IER)

      CALL DFILW (OFLNM2,IER)
      CALL CFILW (OFLNM2,2,KER)
      CALL CHECK(KER)
      OPEN 2, OFLNM2,ATT="OR",ERR=100
100   CONTINUE
5     TYPE" "
      TYPE"ENTER SIZE OF FILES:"
      ACCEPT" 256, 128, 64, OR 32 -->",ISIZE
C
C  TEST FOR ONLY THE ALLOWABLE SIZES
C
      IF(ISIZE.EQ.256) GO TO 6
      IF(ISIZE.EQ.128) GO TO 6
      IF(ISIZE.EQ.64) GO TO 6
      IF(ISIZE.EQ.32) GO TO 6
      GO TO 5
6     CONTINUE   ; size is okay

      IFACTOR=256/ISIZE                  ;find scaling factor
      ITIMES=64/IFACTOR**2-1
```

```
C
C  TRANSFER FILE INTO NEW BINARY FILE
C
       DO 33 I=0,ITIMES

          CALL RDBLK(1,I,TEMP,1,IER)
          CALL CHECK(IER)
          CALL UNPACK(256,TEMP,IN)
          DO 2 J=1,1024
             WRITE BINARY (2) IN(J)
     2    CONTINUE

33     CONTINUE

       CALL RESET
       STOP"<7><7><7><7>BI"
       END
```

```
C*****************************************************************
C                                                                *
C   HISTO.FR  - DG FORTRAN 5 -  LT RICHARD MILLS, 11 SEPT 1984   *
C                                                                *
C       This program will examine the pixel values of a video    *
C       file by producing a histogram of the values.  If a       *
C       Tektronix 4010 graphics terminal is available a graph    *
C       of the histogram may be constructed.                     *
C                                                                *
C     RELOAD LINE:                                               *
C       RLDR HISTO UNPACK @GRAPH@ @FLIB@                         *
C                                                                *
C     COMMAND LINE:                                              *
C       HISTO                                                    *
C                                                                *
C*****************************************************************

      INTEGER TEMP(256),IN(1024)
      INTEGER INFLNM(7)
      REAL HIS(16)

      DO 40 I=1,16        ;SET HIS ARRAY TO ZEROS
         HIS(I)=0.0
 40   CONTINUE
C
C ACCEPT INPUTS
C
      ACCEPT"ENTER  FILENAME ->"
      READ(11,1)INFLNM(1)
    1 FORMAT(S13)

      CALL OPEN(1,INFLNM,1,IER)
      CALL CHECK(IER)

5     TYPE" "
      TYPE"ENTER SIZE OF FILE:"
      ACCEPT" 256, 128, 64, OR 32 -->",ISIZE
C
C  TEST FOR ONLY THE ALLOWABLE SIZES
C
      IF(ISIZE.EQ.256) GO TO 6
      IF(ISIZE.EQ.128) GO TO 6
      IF(ISIZE.EQ.64) GO TO 6
      IF(ISIZE.EQ.32) GO TO 6
      GO TO 5
6     CONTINUE   ; size is okay

      IFACTOR=256/ISIZE                    ;calculate scaling factors
      ITIMES=64/IFACTOR**2-1
```

```
C
C READ FILE VALUES
C
      DO 3 I=0,ITIMES

         CALL RDBLK(1,I,TEMP,1,IER)
         CALL CHECK(IER)
         CALL UNPACK(256,TEMP,IN)

         DO 2 J=1,1024
            HIS(IN(J)+1)=HIS(IN(J)+1)+1.0
    2    CONTINUE

    3 CONTINUE
C
C TYPE HISTOGRAM INFORMATION
C
      DO 50 I=1,16
        IOFF=I-1
        TYPE"VALUE:",IOFF,"  NUMBER:",HIS(I)
50    CONTINUE

      ACCEPT"ENTER 1 FOR A GRAPH ",ITOGGLE
      IF(ITOGGLE.NE.1)GO TO 51
C
C PLOT THE GRAPH
C
      CALL GRPH2("HISTOGRAM OF PIXEL VALUES",1,HIS,U,16,1,YMIN,
     +YMAX,0)
      ACCEPT DUMMY  ;KEEP THE PROMPT OFF SCREEN FOR THE COPYING

51    CALL RESET
      STOP
      END
```

```
C****************************************************************
C                                                               *
C    ORIGINAL PROGRAM: NEG.FR BY R. SIMMONS                     *
C    REWRITTEN: BY R. MILLS                                     *
C    This program will create the "negative" of a video frame.  *
C    Used for an alternate persective on the hard copies and    *
C    to conserve lineprinter ink.                               *
C                                                               *
C       The command line format is:                             *
C                                                               *
C                  NEG                                          *
C                                                               *
C****************************************************************

        DIMENSION IBLKS(16384),NEG(16)
        ACCEPT "ENTER POSITIVE FILENAME: "
        READ (11,5) POS
 5      FORMAT (S40)
        ACCEPT "ENTER NEGATIVE FILENAME: "
        READ (11,5) NEG(1)
        OPEN 1,POS   ; [illegible] positive file on ch. 1
        CALL DFILW (N[illegible] ,IER)
        IF(IER.NE.1)GOTO 90
90      CALL CFILW(NEG, 3,64,KER)      ;not exist on the disk
        CALL CHECK(KER)
        OPEN 2,NEG,ATT="OC",ERR=100
C
 100    CALL RDBLK(1,0,IBLKS,64,MER)  ;Read 64 blocks into IBLKS
        IF(MER.NE.1)TYPE" RDBLK error:",MER
        TYPE "WORKING"
        DO 1 J=1,16384
   1       IBLKS(J)=NOT(IBLKS(J))      ;Form the negative image
        CALL WRBLK(2,0,IBLKS,64,NER)  ;Write the blocks
        IF(NER.NE.1)TYPE" WRBLK error:",NER
C
        TYPE "FINISHED!"
        CALL RESET                     ;Close all channels
        STOP
        END
```

```
      SUBROUTINE UNPACK(N,PIXWORD,PIXELS)
C
C     Written by Lt. Simmons(Ref 9)          Version 2
C
C    This subroutine will unpack four 4-bit integers from a
C    16-bit integer word.  The pixels in a video file have to
C    be unpacked if each pixel is to be operated on separately.
C
      INTEGER PIXWORD(N),PIXELS(4,N)     ;Four pixels per word
      DO 1 I=1,N                         ;'N' allows higher-order
      DO 1 J=1,4                         ;arrays to be passed.
      PIXELS((5-J),I)=15.AND.PIXWORD(I);Pick off right pixel
    1 PIXWORD(I)=ISHFT(PIXWORD(I),-4)    ;Shift word 4 bits right
      RETURN                             ;to pick off next pixel.
      END
      SUBROUTINE REPACK(N,PIXELS,PXWD)
C
C     Written by Lt. Simmons (Ref 9)          Version 2
C
C     This subroutine will repack four 4-bit integer pixels
C     into one 16-bit word for use by CHOPS.  Parameter N
C     allows more than one 4-bit to 1-word repacking
C     operation in each call to REPACK.
C
      INTEGER PIXELS(4,N),PXWD(N)
      DO 1 J=1,N                        ;Loop N times
      PXWD(J)=0
      DO 1 I=1,4
      PXWD(J)=ISHFT(PXWD(J),4)      ;Shift pixel left in word
    1 PXWD(J)=PIXELS(I,J)+PXWD(J) ;then add next pixel on right
      RETURN
      END
```

APPENDIX E

Test Cases

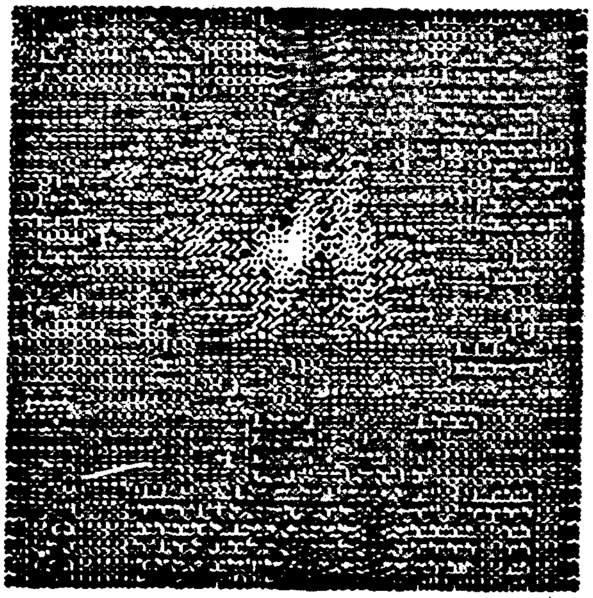
This Appendix contains the images and results of additional test cases operated on by the described algorithm. Each case has a brief description explaning the type of scene used and the resulting images of the scene anaylsis.

Six test cases ,II-VII, are included in this Appendix in addition to Test Case I which is covered in the text of this thesis.

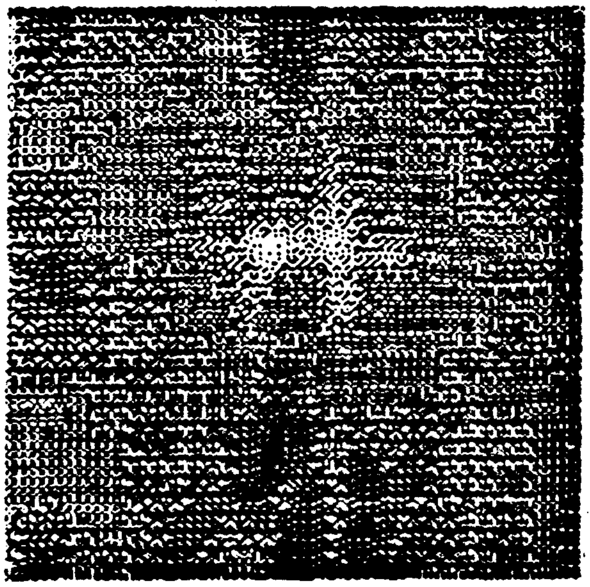
Test Case II

This test case deals with locating the F-16's in Figure 2.2 with the normal, generalized F-16 template of Figure 3.7b. The Wedging has already been presented in the text in Figure 2.7. The first pass correlation, using the 39x39 pixel filter, was also presented in Figure 3.11. From this correlation output the same five potential targets of Figure 4.2 were extracted from the edge detected image, energy normalized and reccorelated with the template of Figure 3.7b.

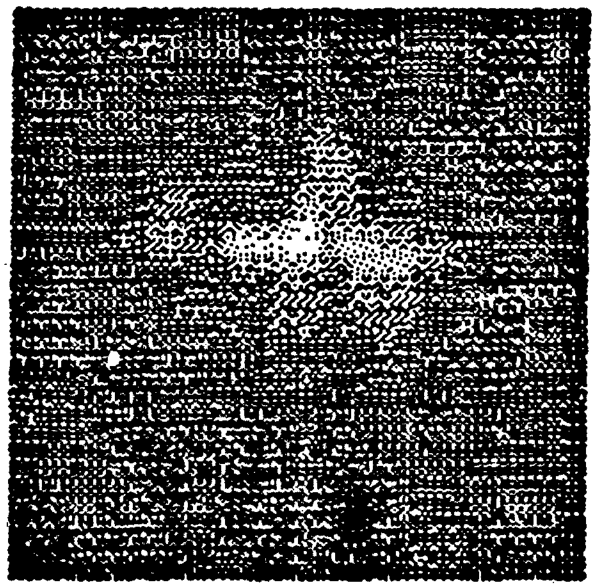
Figure E.1 shows the output of the recorrelation of these smaller, energy normalized images. The evaluation of the peaks within these images are included in Table II. Note that rating for the 4 actual targets is rather low, but it is higher than the rating for the image of the false target.



a

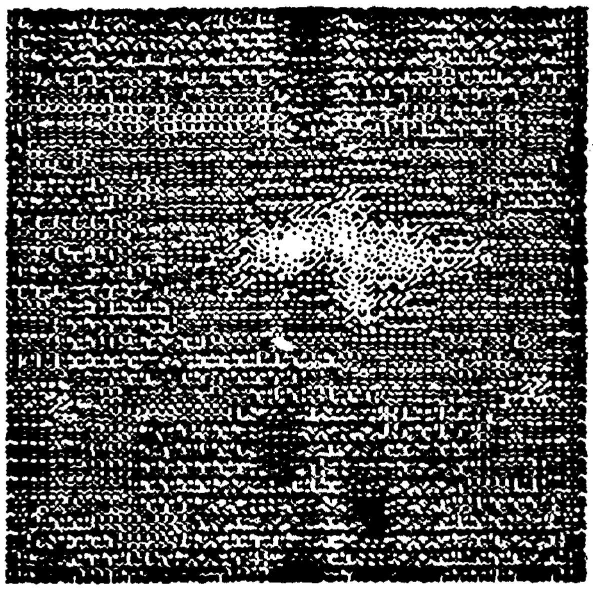


b



c

Figure E.1a-c Second Pass Correlation Output of Case II.
a) Correlation Output of Figure 4.2a with General F-16 Template of Figure 3.7b,
b) Correlation Output of Figure 4.2b with General F-16 Template of Figure 3.7b,
c) Correlation Output of Figure 4.2c with General F-16 Template of Figure 3.7b.
(No reduction of PIX output)


d

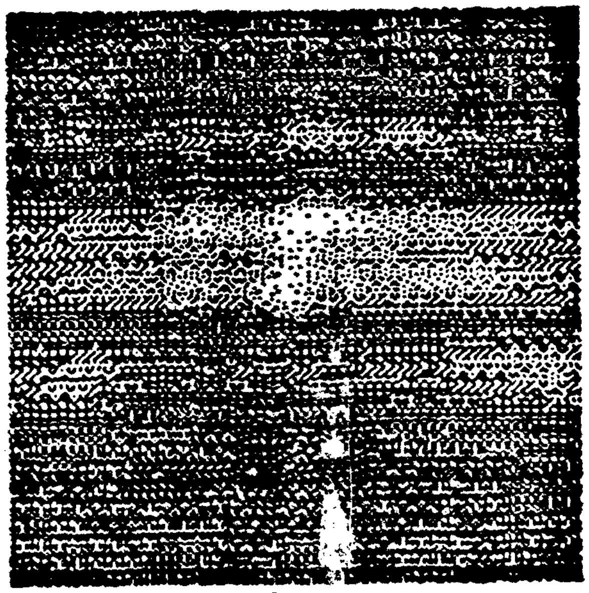

e

Figure E.1d-e Second Pass Correlation Output of Case II.
d) Correlation Output of Figure 4.2d with General F-16 Template of Figure 3.7b.
e) Correlation Output of Figure 4.2e with General F-16 Template of Figure 3.7b.
(No reduction of PIX output)

Table II

Evaluation of Test Case II

EVALUATION RESULTS
SCENE FILENAME: FIGURE E.1

FILE NAME	SCORE1	SCORE2	TOTAL SCORE	RANK
FIGE1A.VD	.796275	.742080	.590900	6
FIGE1B.VD	.811202	.783621	.635676	7
FIGE1C.VD	.796275	.715959	.570100	6
FIGE1D.VD	.827416	.725781	.600523	7
FIGE1E.VD	.733916	.566479	.415748	5

Test Case III

The image of three T-38's, shown in Figure E.2, is considered in this third test case. The edge extraction is shown in Figure E.3. Figure E.4 contains the templates used for the first and the second pass correlation. The second, smaller template was used to keep the image sizes within 64x64 pixels.

The output of the first pass correlation is given in Figure E.5 and the smaller, potential targets corresponding to the peaks are shown in Figure E.6. The second pass correlation output is shown in Figure E.7 and the evaluation of these peaks is presented in Table III.

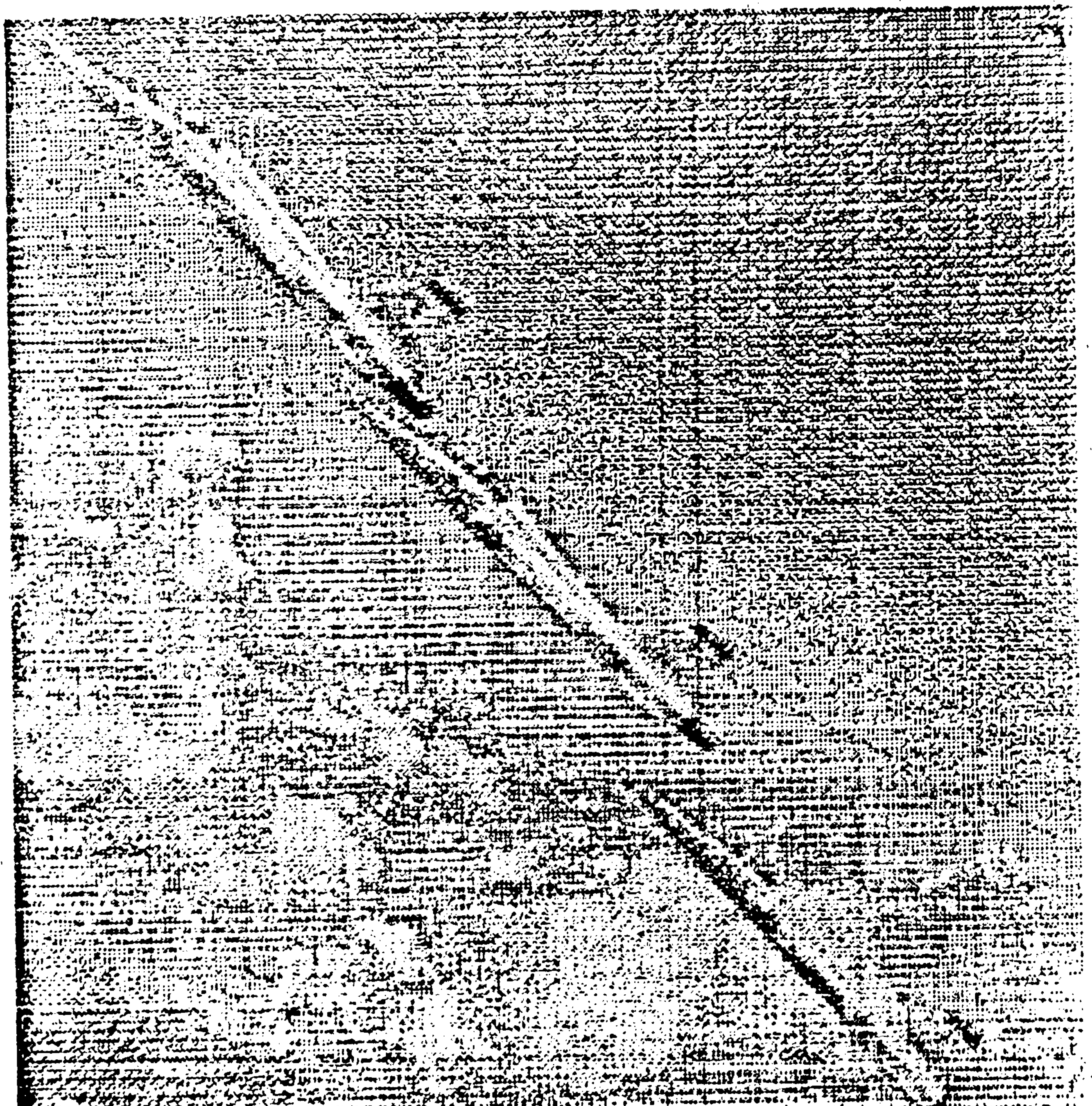


Figure E.2 Input Image of Test Case III.

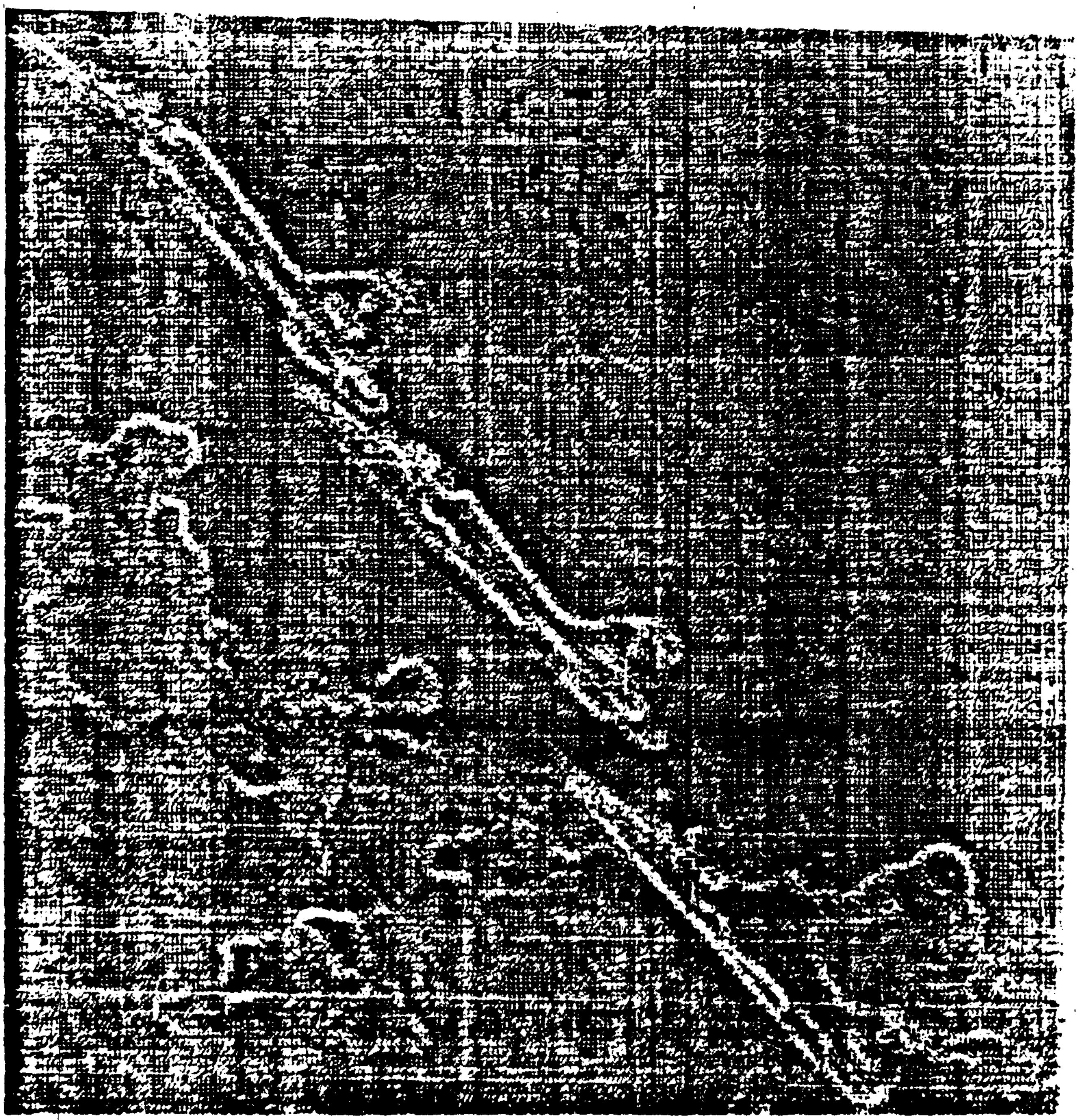


Figure E.3 Edge Extraction of Figure E.2.

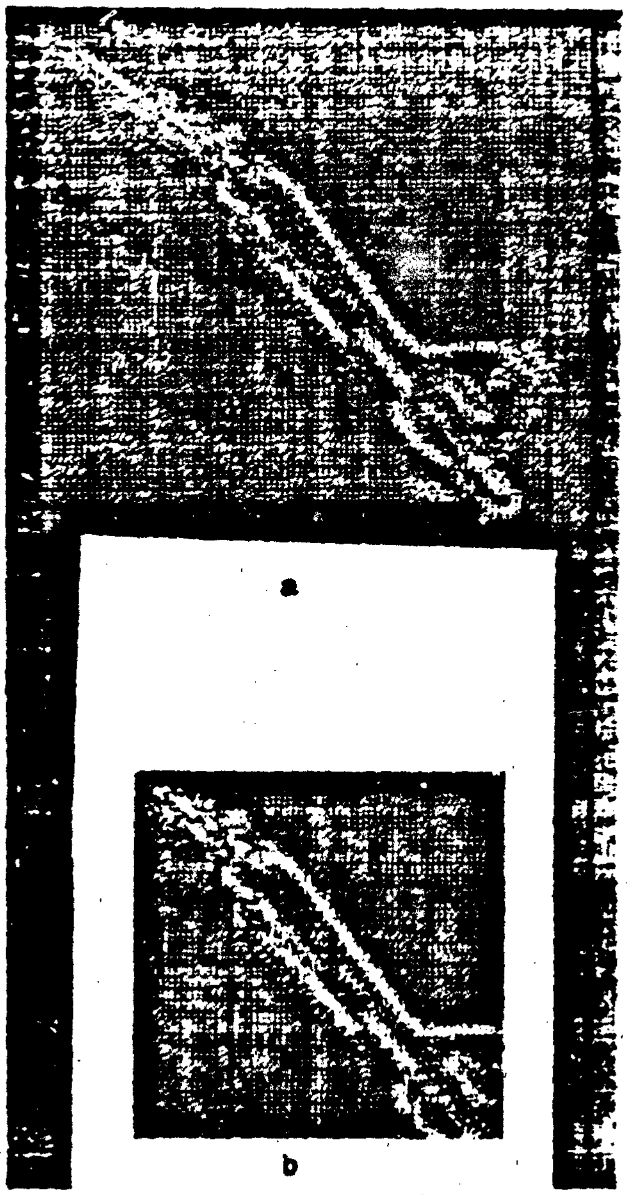


Figure E.4 T-38 Templates.
a) Template for First Pass Correlation
b) Template for Second Pass Correlation.

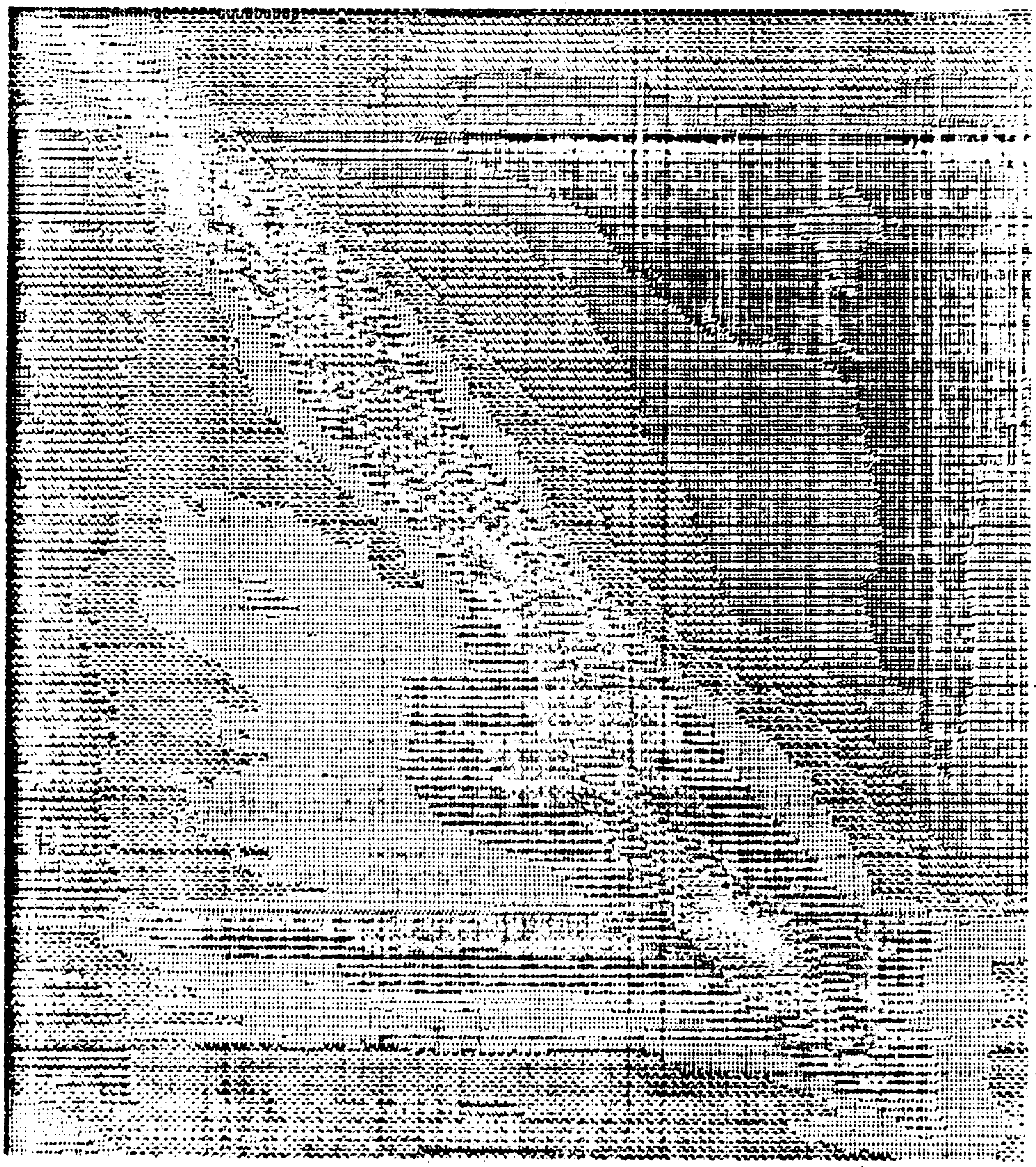


Figure E.5 First Pass Correlation Output of Test Case III.

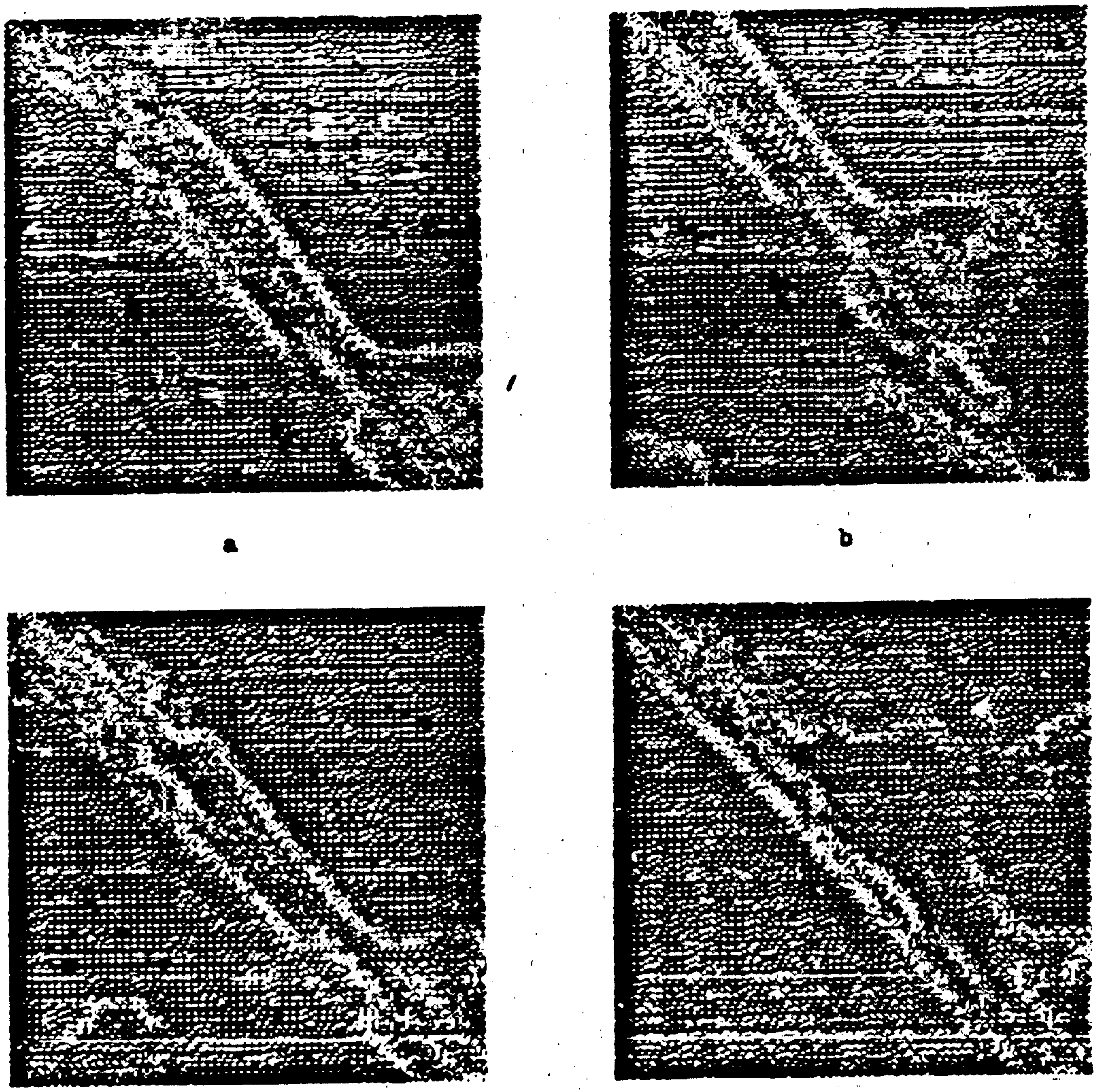
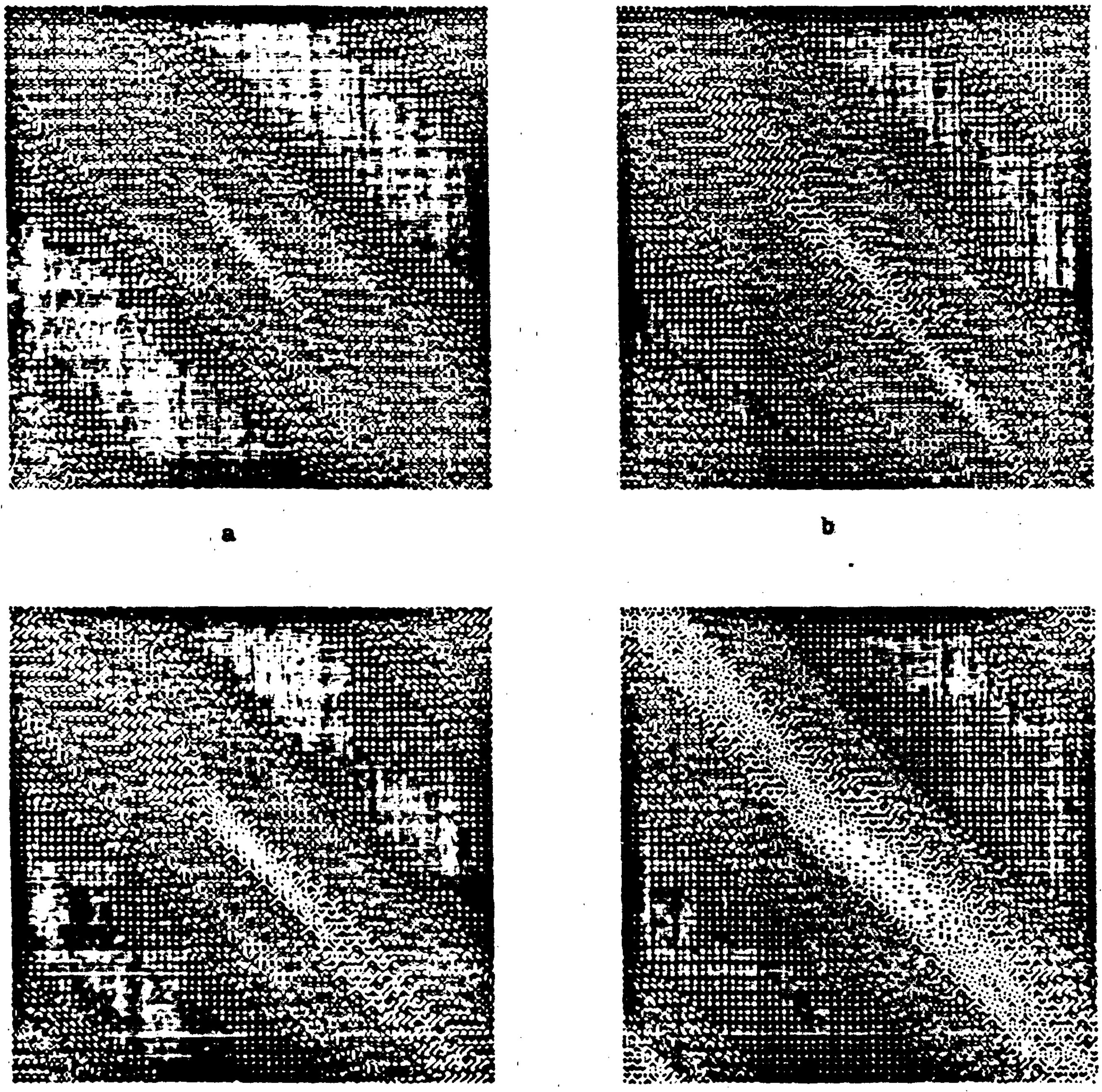

a b c d

Figure E.6 Small Files Corresponding to Peaks in Figure E.5. (No reduction of PIX output)



a b c d

Figure E.7 Second Pass Correlation Output of Test Case III.
(No reduction of PIX output)

Table III

Evaluation of Test Case III

```
                      EVALUATION RESULTS
                  SCENE FILENAME:  FIGURE E.7
FILE NAME          SCORE1       SCORE2      TOTAL SCORE      RANK
---------          ------       ------      -----------      ----
FIGE7A.VD          .977903      .839515       .820964          9
FIGE7B.VD          .240619      .731477       .176007          2
  POSSIBLE OFF-CENTERED PARTIAL TARGET
FIGE7C.VD          .977903      .710138       .694446          7
FIGE7D.VD          .977903      .494797       .483364          5
```

Test Case IV

This fourth test case examines the image of five F-86's shown in Figure E.8. The image is relatively free from clutter except for the contrails which show up very well after edge extraction as shown in Figure E.9. Figure E.10 is the target template used to produce the first pass correlation output of Figure E.11. The small potential targets are shown in Figure E.12 and their second pass, energy normalized correlations are presented in Figure E.13. The evaluation of these eight recorrelations is given in Table IV.

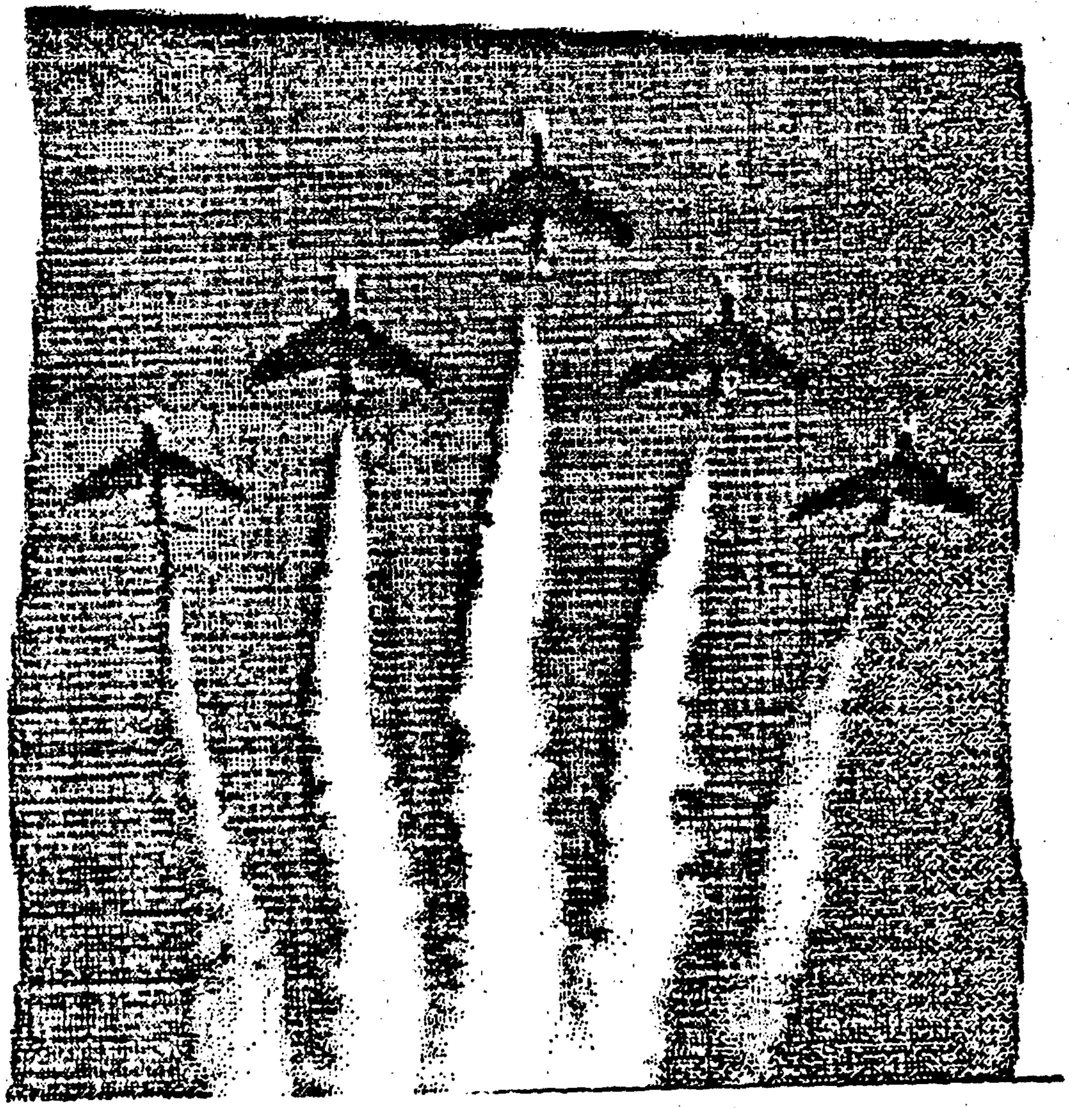


Figure E.8 Input Image of Test Case IV.

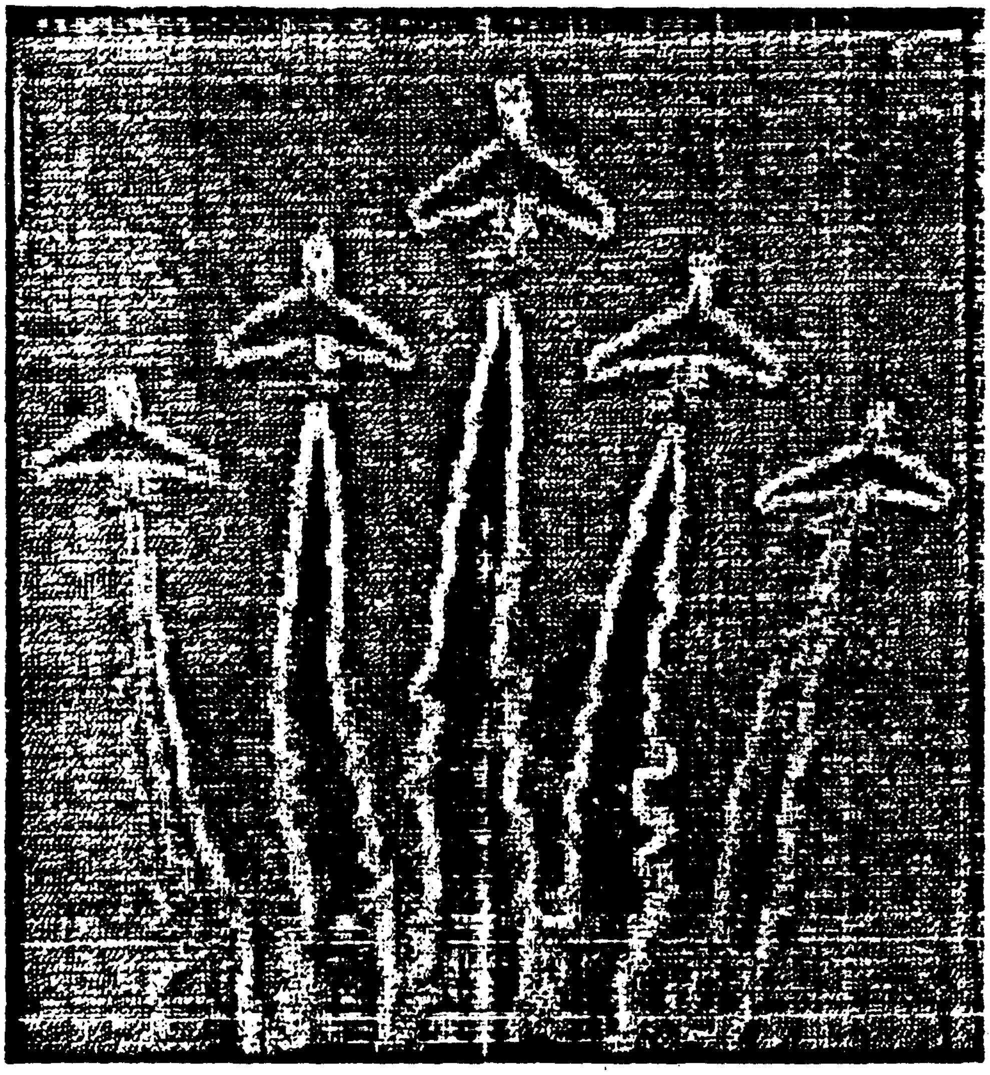


Figure E.9 Edge Extraction of Figure E.8.

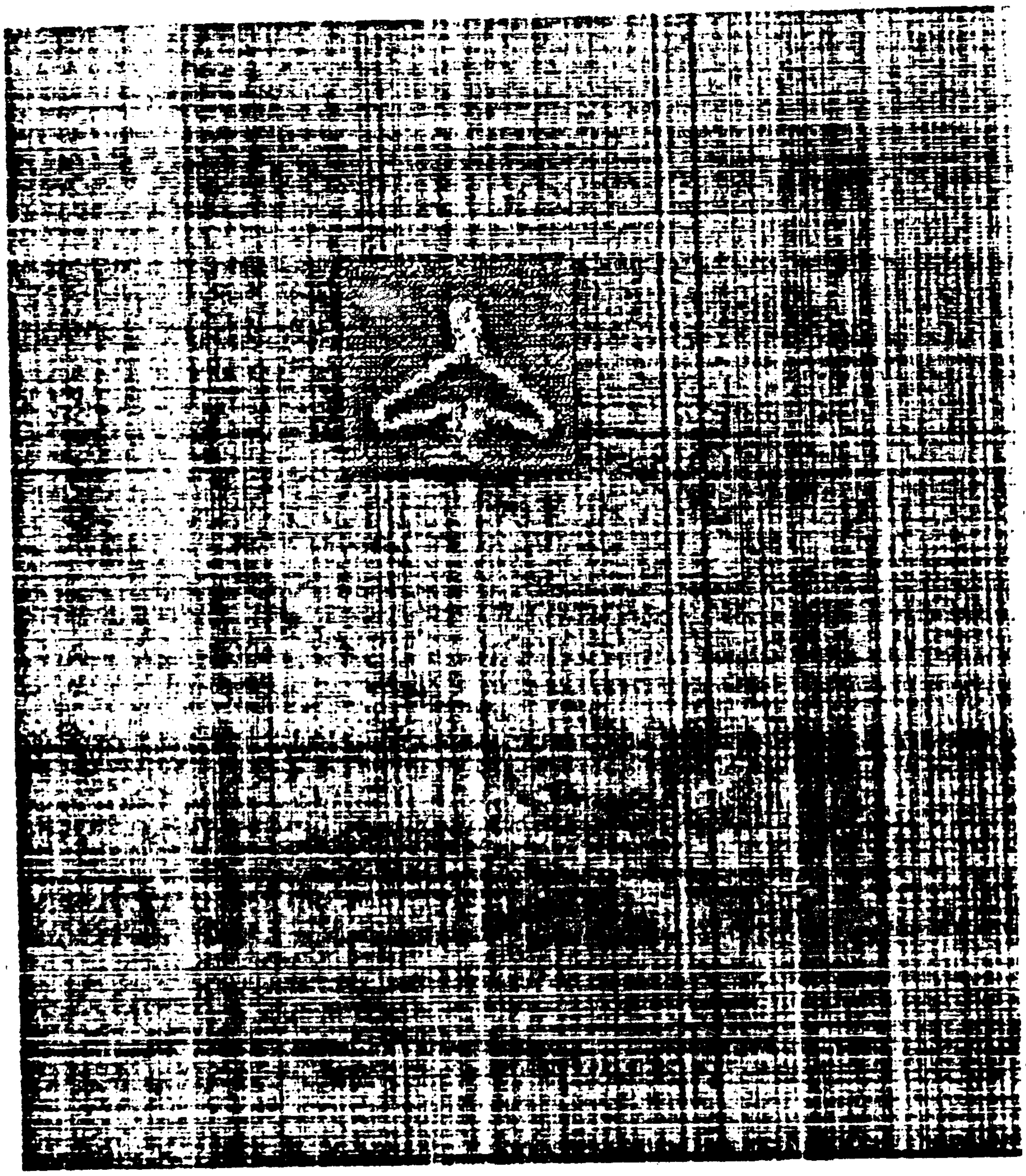


Figure E.10 F-86 Template.

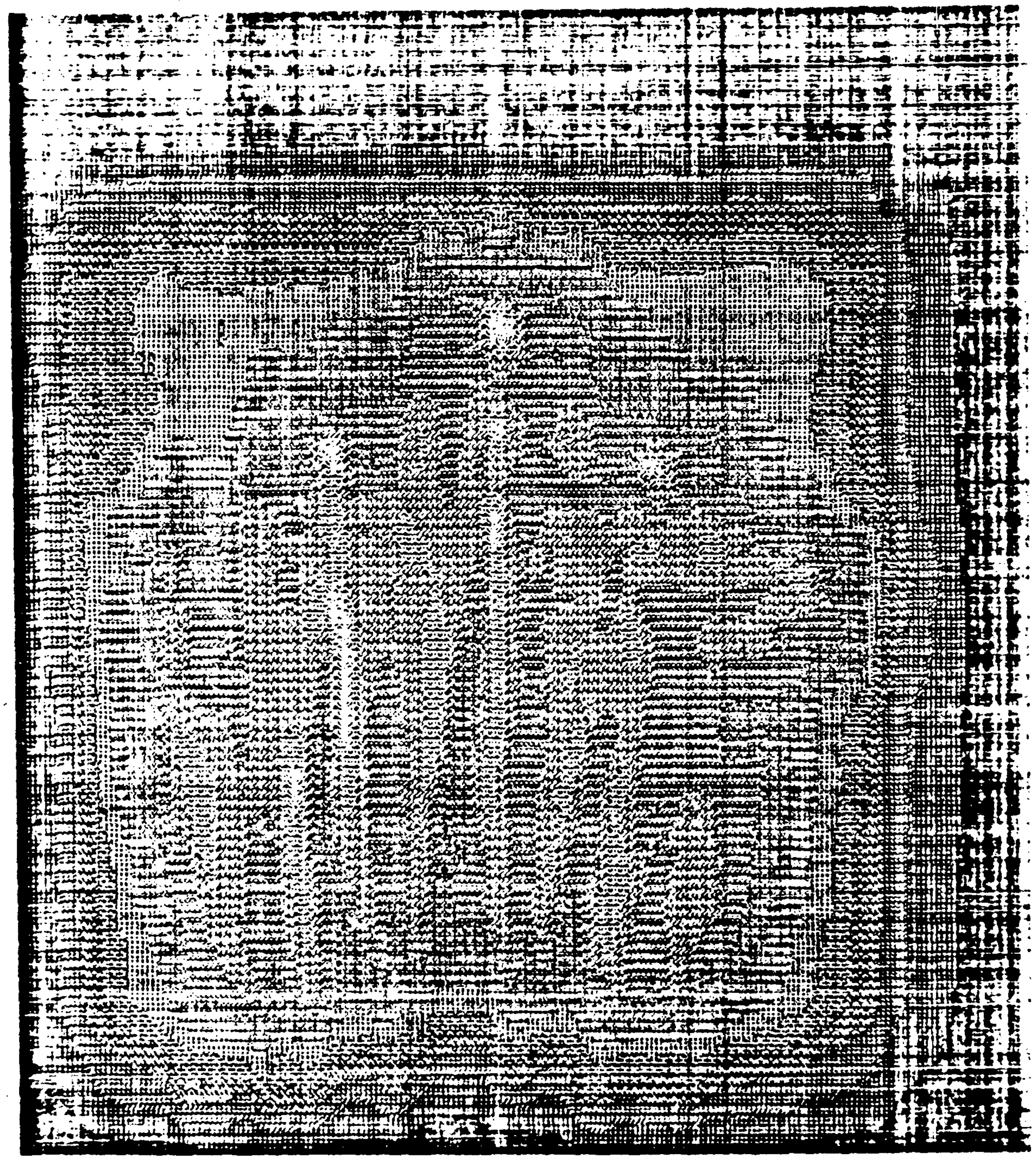
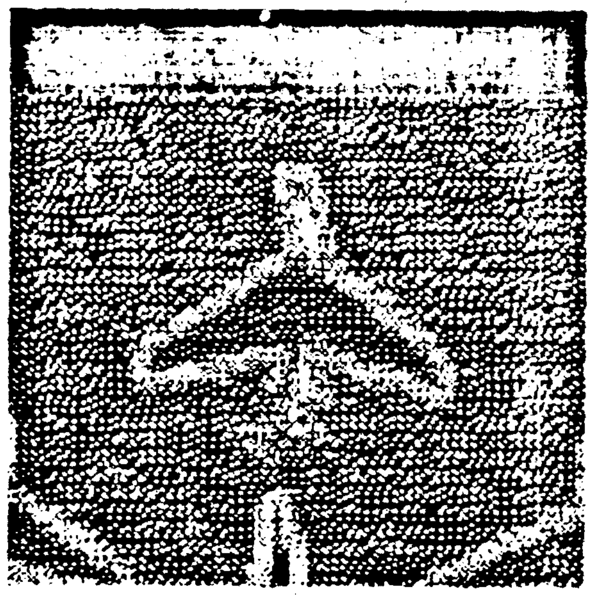
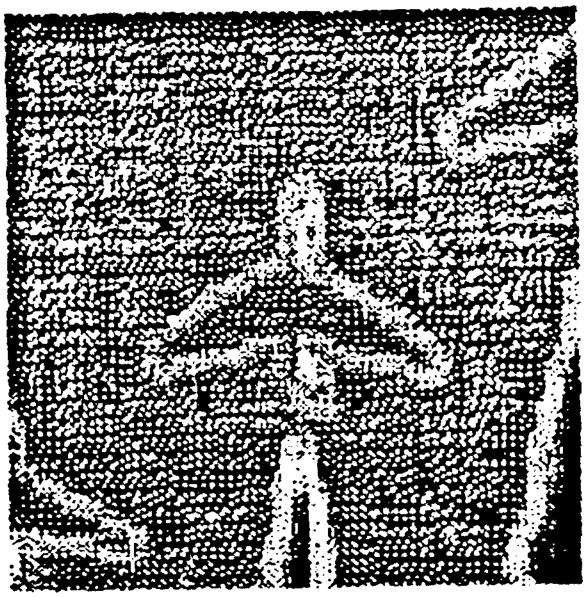
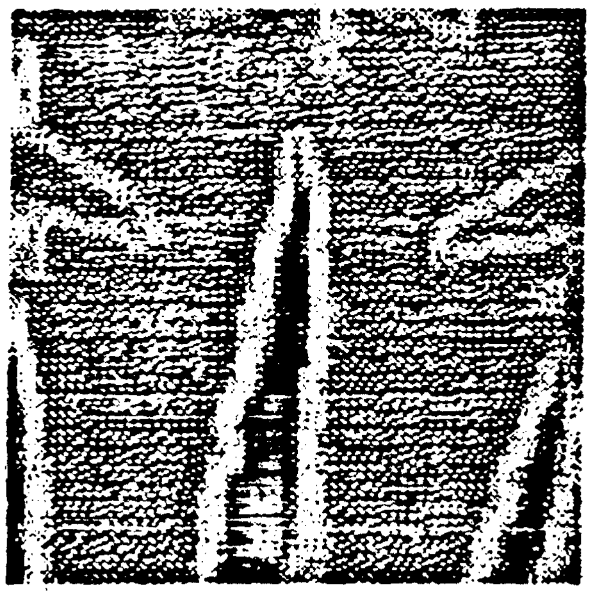


Figure E.11 First Pass Correlation Output of Test Case IV.


a


b


c

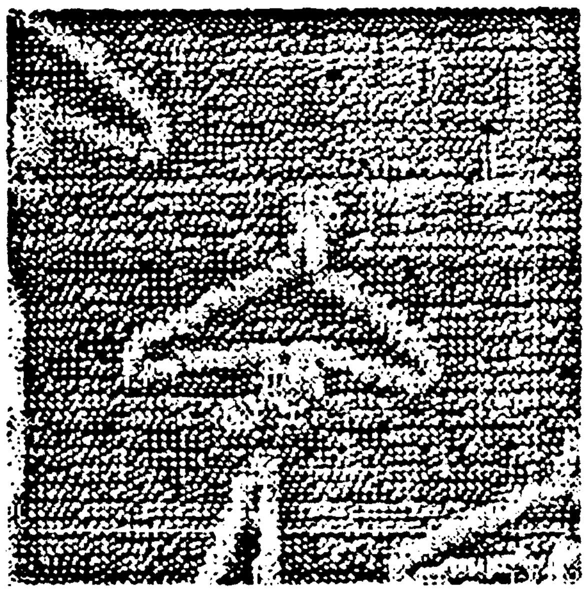
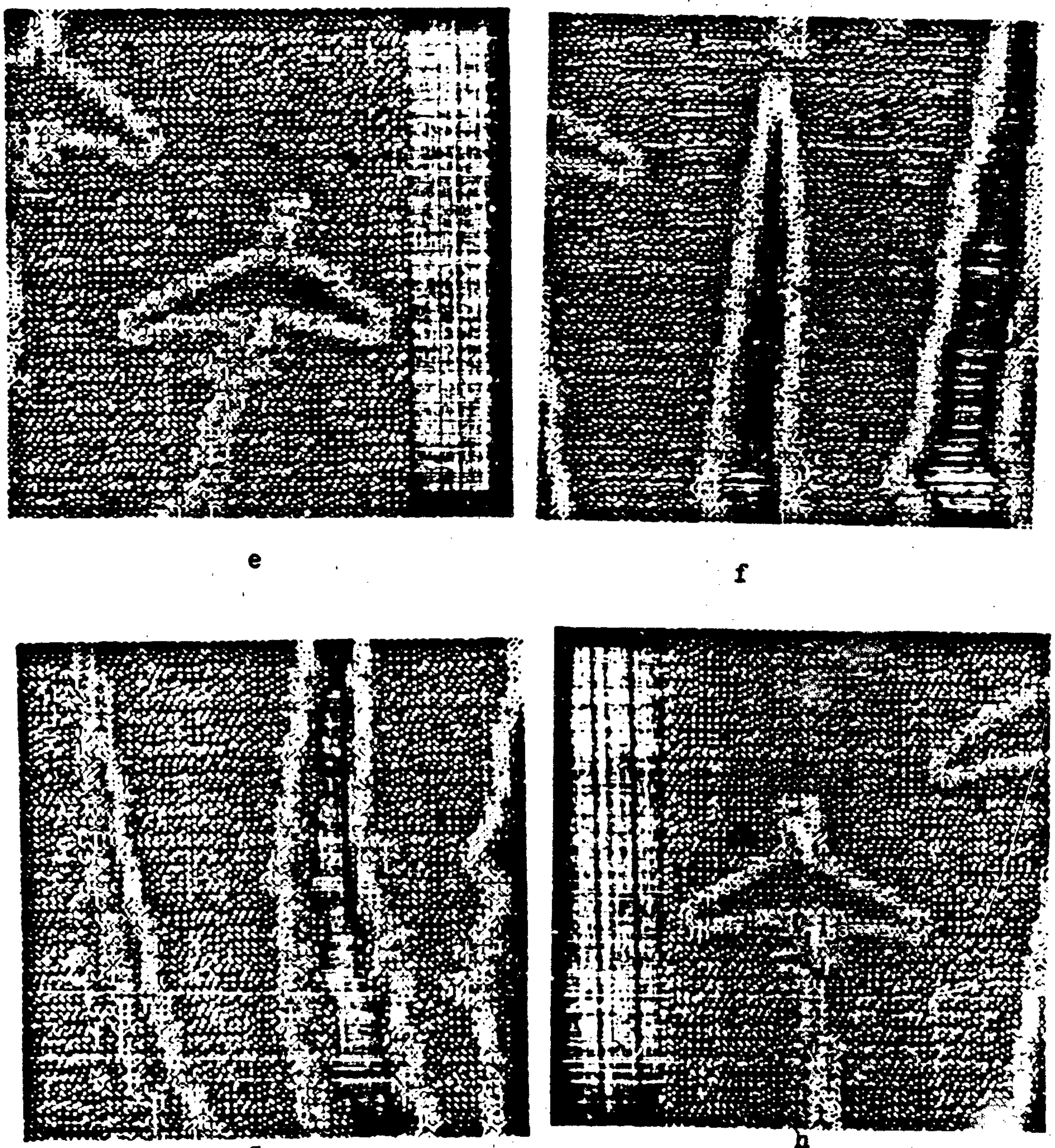

d

Figure E.12a-d Small Files Corresponding to Peaks in Figure E.11. (No reduction of PIX output)


e

f

g

h

Figure E.12e-h Small Files Corresponding to Peaks in Figure E.11. (No reduction of PIX output)

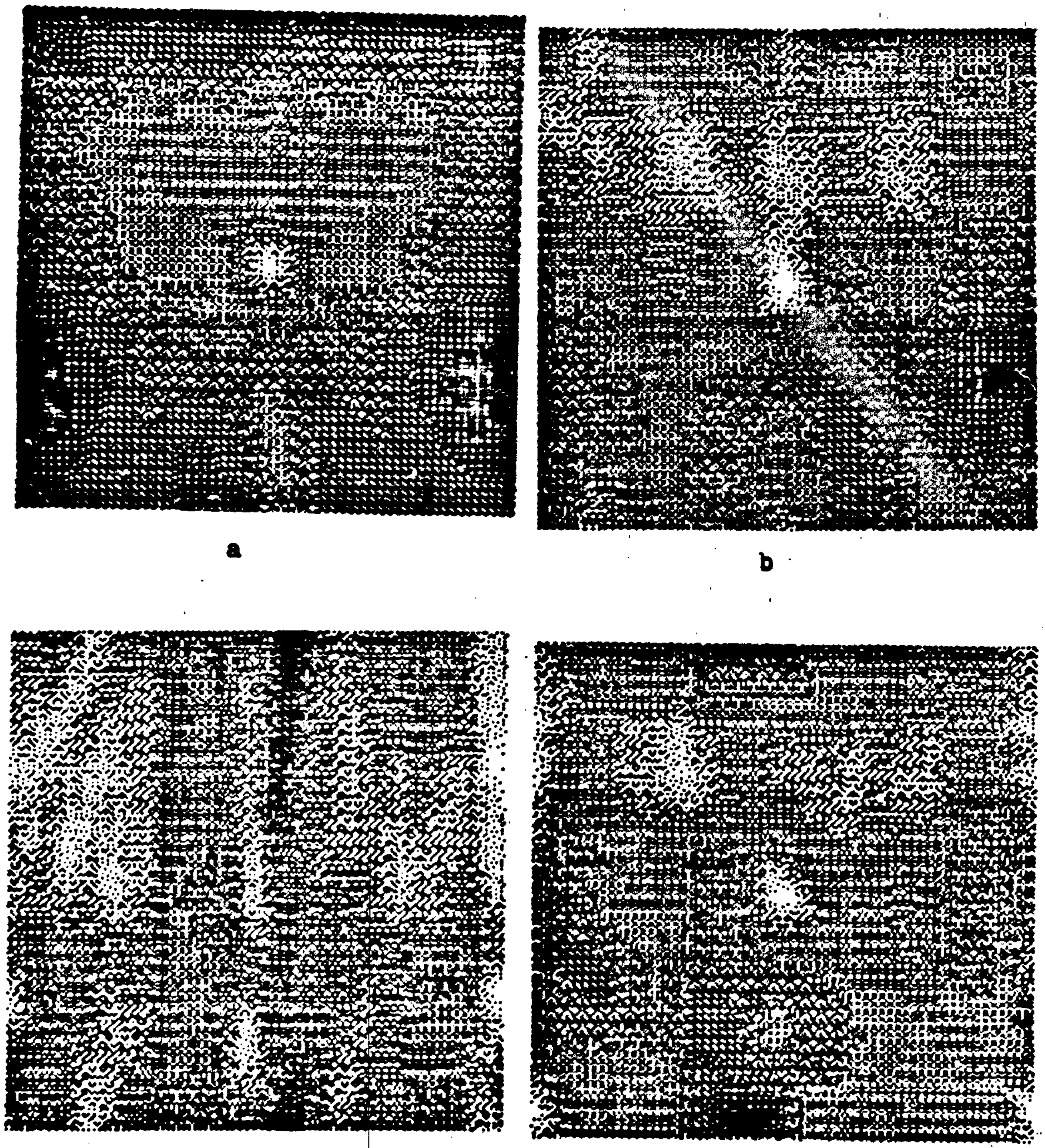


Figure E.13a-d Second Pass Correlation Output of Test Case IV.
(No reduction of PIX output)

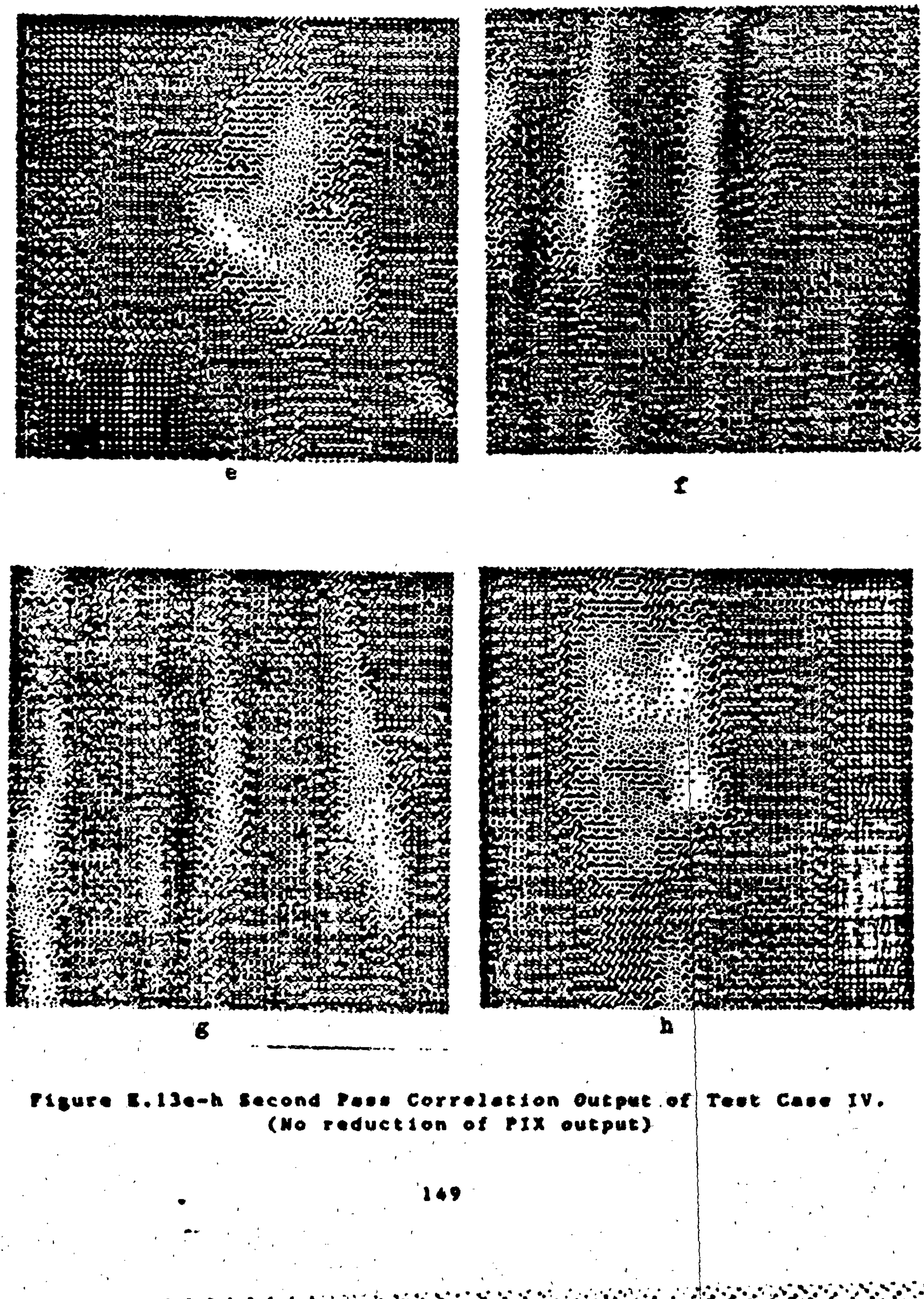

e f g h

Figure E.13e-h Second Pass Correlation Output of Test Case IV.
(No reduction of PIX output)

Table IV

Evaluation of Test Case IV

EVALUATION RESULTS
SCENE FILENAME: FIGURE E.13

FILE NAME	SCORE1	SCORE2	TOTAL SCORE	RANK
FIGE13A.VD	.977903	.899001	.879136	9
FIGE13B.VD	.977903	.670238	.655427	7
FIGE13C.VD	.000000	.581368	.000000	1
FIGE13D.VD	.000000	.656622	.000000	1
POSSIBLE OFF-CENTERED PARTIAL TARGET				
FIGE13E.VD	.977903	.563454	.551003	6
FIGE13F.VD	.445608	.550302	.245329	3
FIGE13G.VD	.139348	.490834	.068396	1
FIGE13H.VD	.977903	.489838	.479014	5

Test Case V

This fifth test case examines the image of four B-52's shown in Figure E.14. The image was aritfically created using a scene generating program by Lt J. Tilley, which superimposed various aircraft images onto a noisy cloud background. The edge extraction of the scene is shown in Figure E.15. Figure E.16 is the edged target template used to produce the first pass correlation output of Figure E.17. The small potential targets, corresponding to the correlation peaks, are shown in Figure E.18. The second pass, energy normalized correlations are shown in Figure E.19. The evaluation of these four recorrelations is presented in Table V.

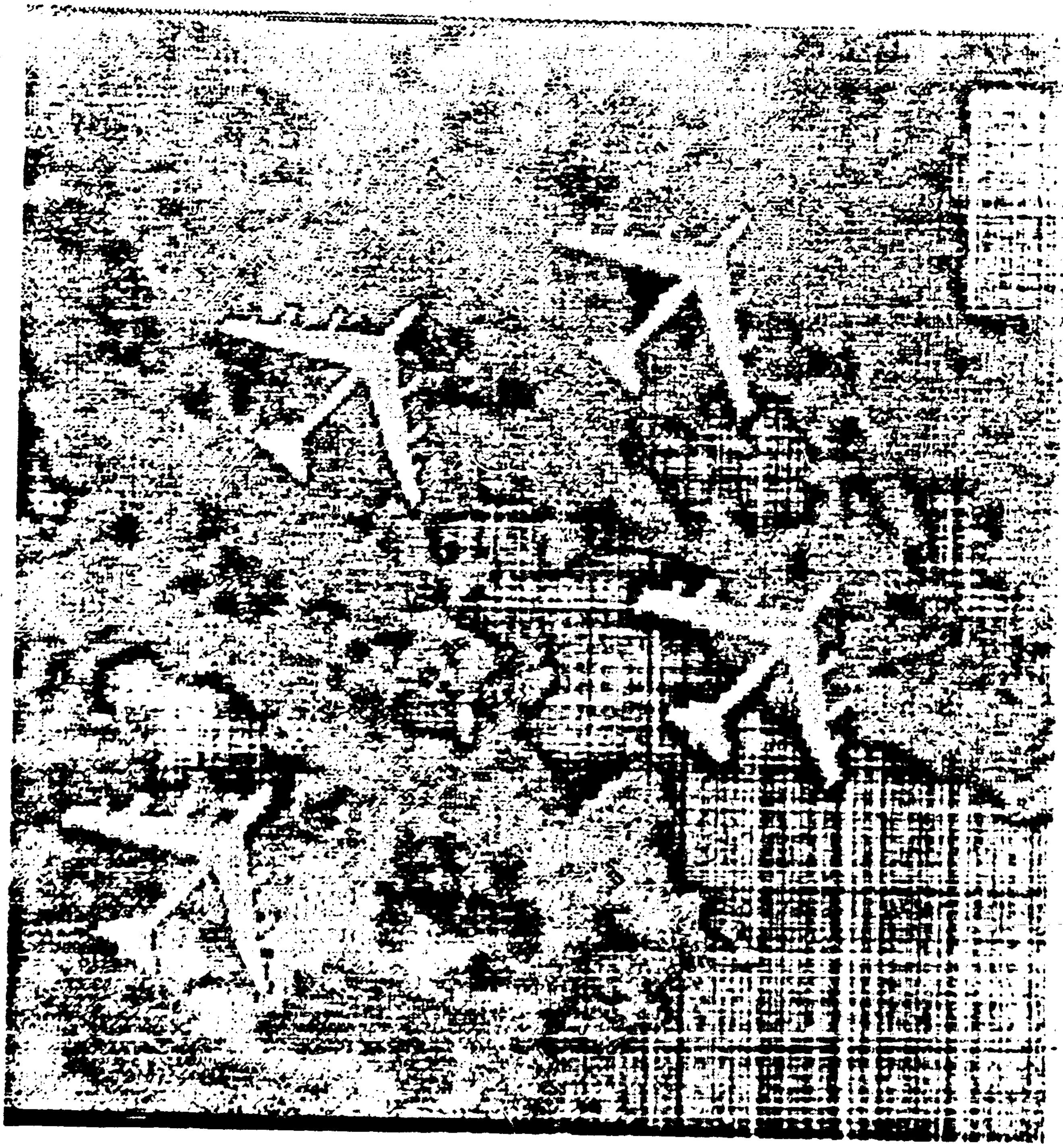


Figure E.14 Input Image of Test Case V.

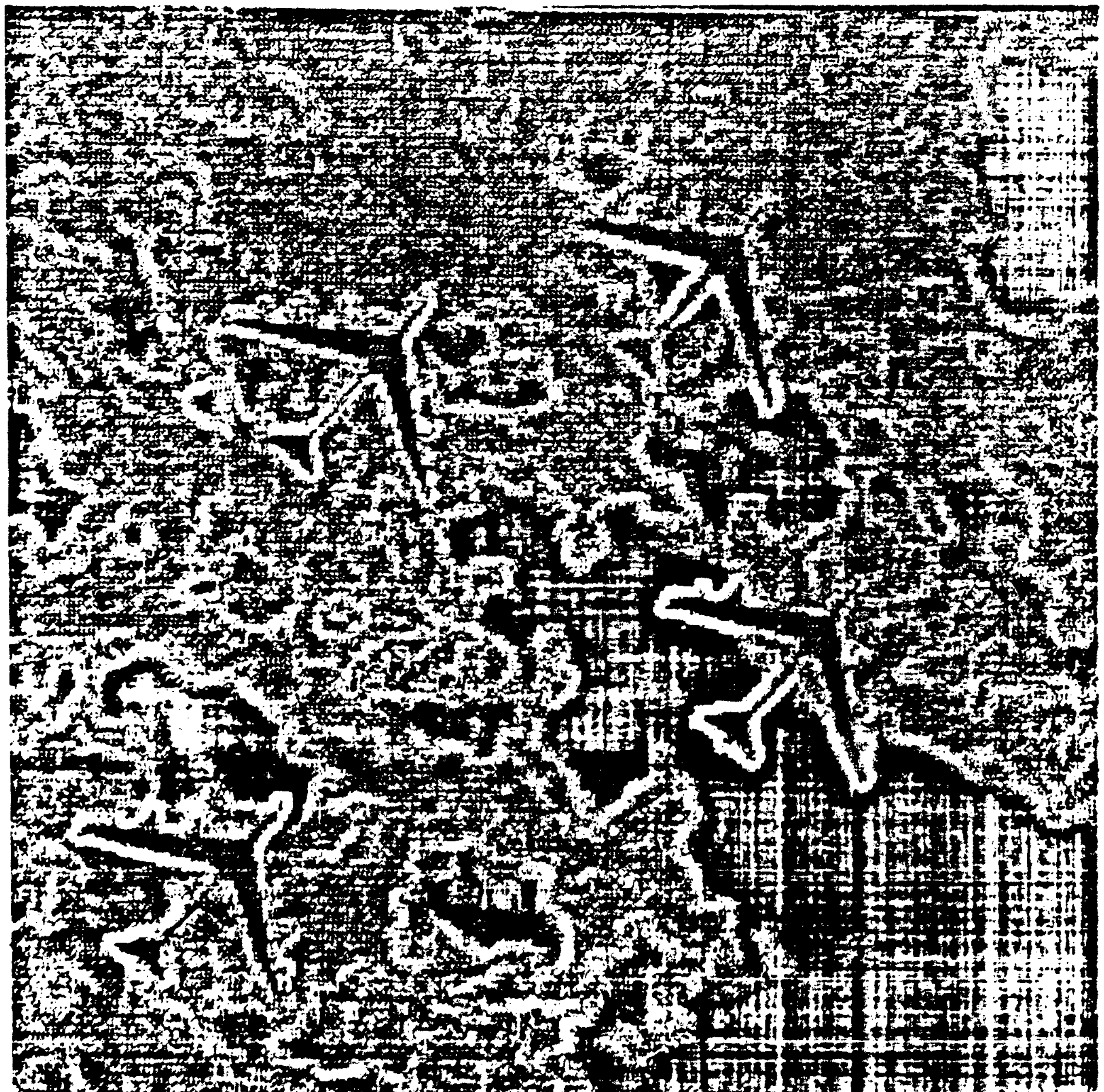


Figure E.15 Edge Extraction of Figure E.14.

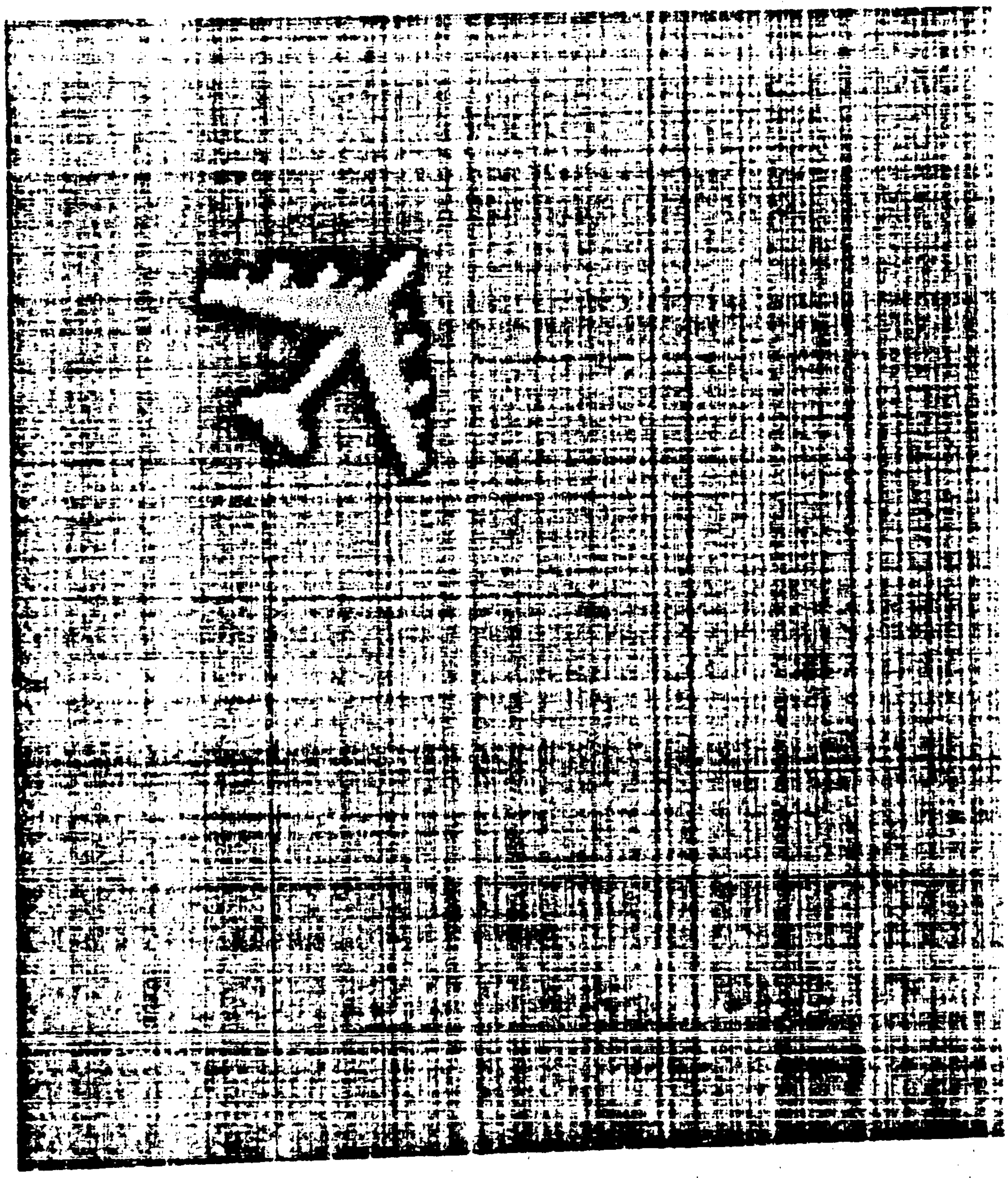


Figure E.16 B-52 Template.

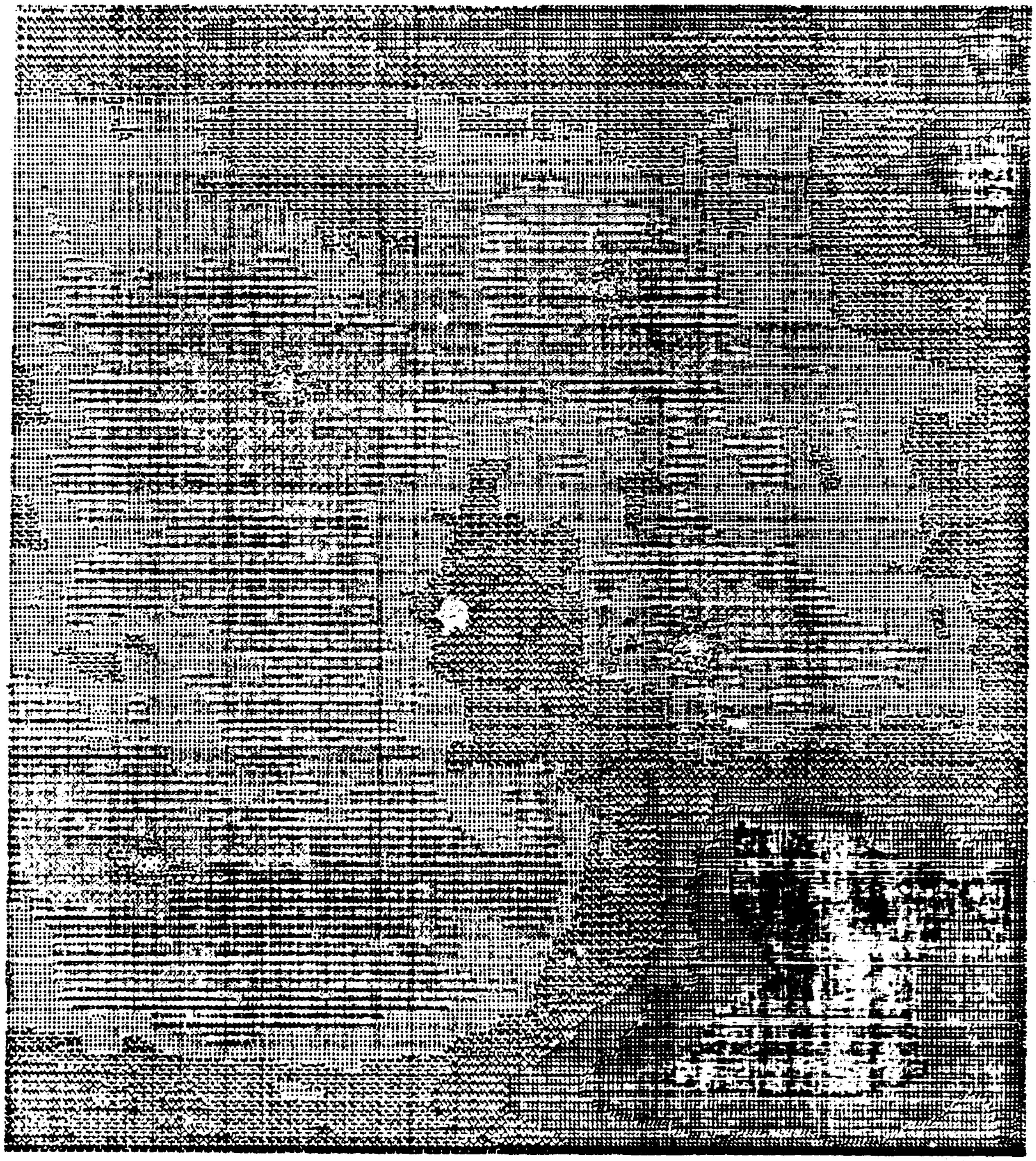
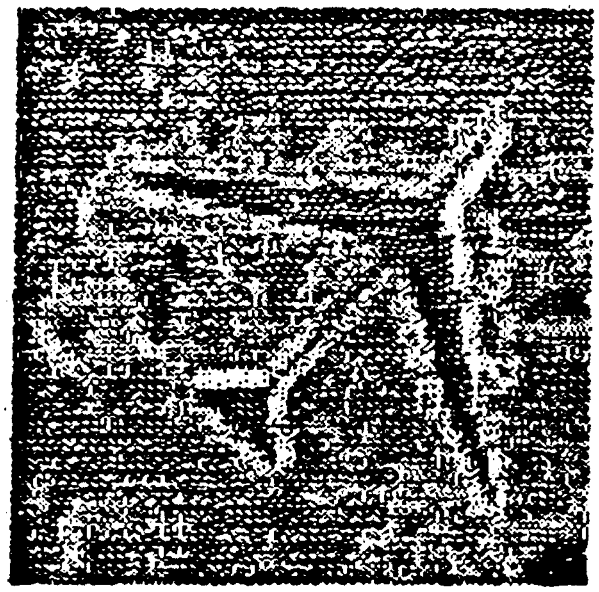
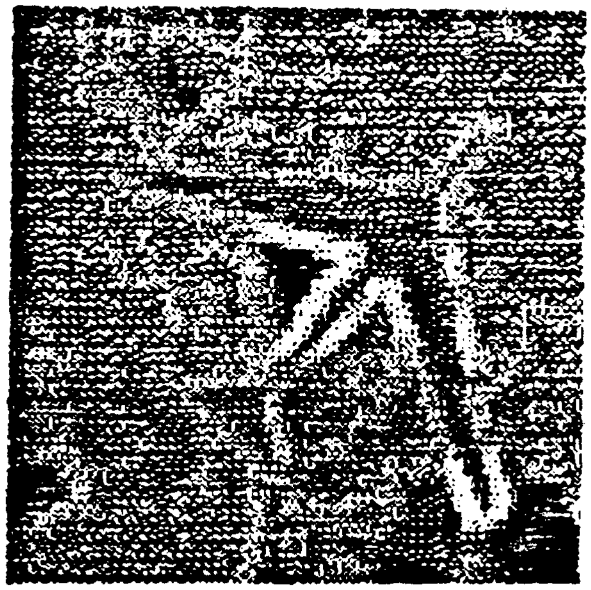


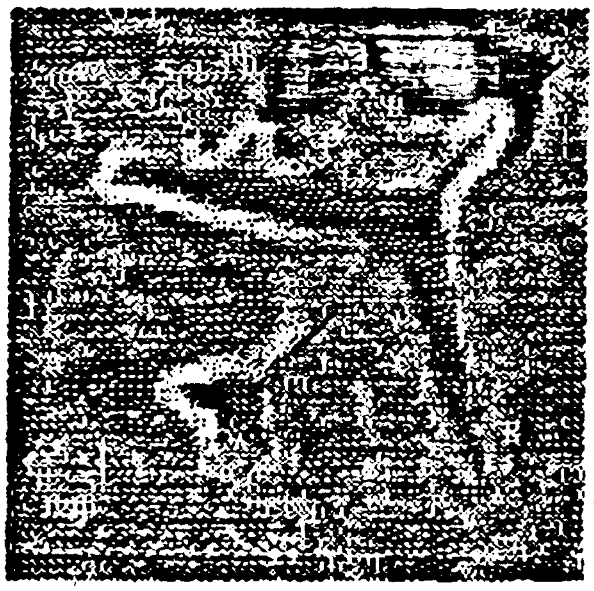
Figure E.17 First Pass Correlation Output of Test Case V.



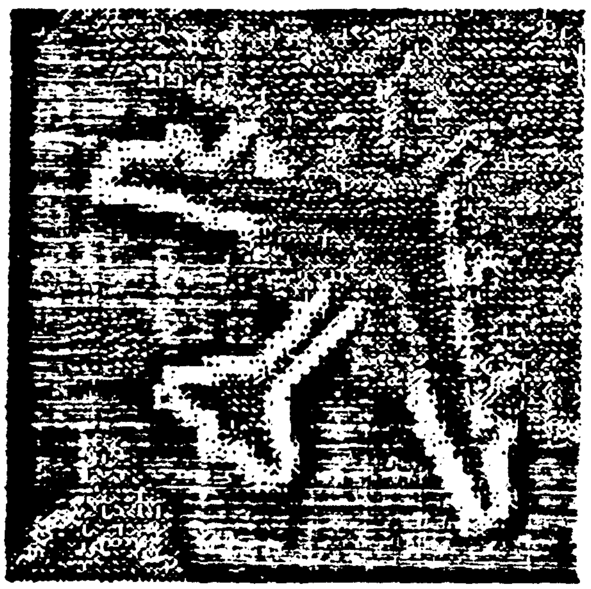
a



b



c



d

Figure E.18 Small Files Corresponding to Peaks in Figure E.17. (No reduction of PIX output)

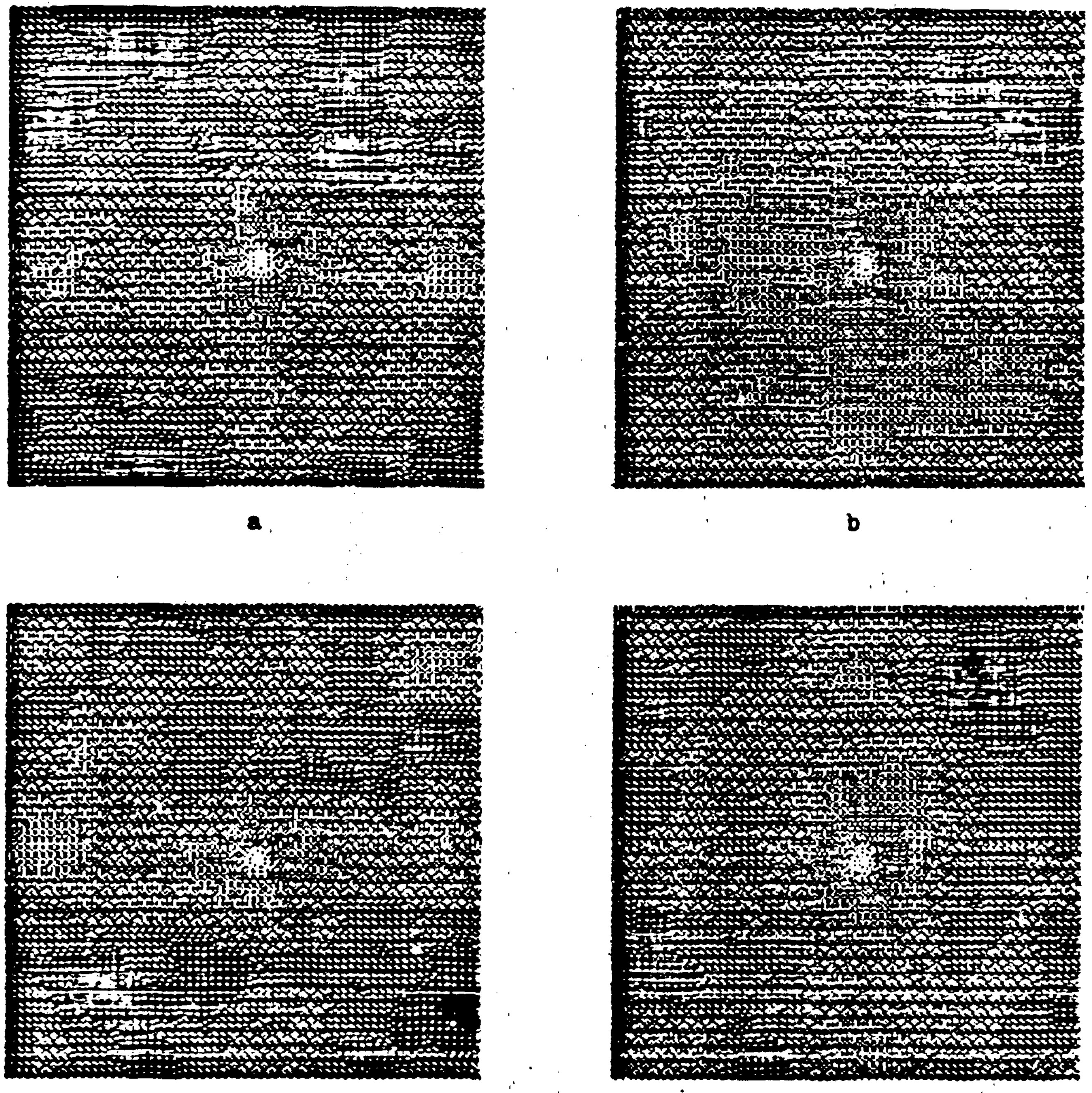

a b c d

Figure E.19 Second Pass Correlation Output of Test Case V.
(No reduction of PIX output)

Table V

Evaluation of Test Case V

EVALUATION RESULTS
SCENE FILENAME: FIGURE E.19

FILE NAME	SCORE1	SCORE2	TOTAL SCORE	RANK
FIGE19A.VD	.889515	.978030	.869972	9
FIGE19B.VD	.889515	.920735	.819008	9
FIGE19C.VD	.889515	.984499	.875726	9
FIGE19D.VD	.889515	.970009	.862837	9

Test Case VI

The image of two B-52's and two KC-135's on a cluttered background, shown in Figure E.20, is examined in Test Case VI. This image was also aritfically produced. The edge extraction of the scene is shown in Figure E.21. Note that the B-52 targets were hidden in the clouds while the KC-135 decoys were placed in relatively clear view. Although this thesis does not emphasize the ability to distinguish the difference between different types of targets, this test case was used to see what the algorithm's performance was in such a situation.

The edged target template used in this test case was the same B-52 template in Figure E.16. The first pass correlation output is shown in Figure E.22. The small potential targets, corresponding to the correlation peaks, are shown in Figure E.23. The second pass, energy normalized correlations are shown in Figure E.24. The evaluation of these four recorrelations is presented in Table VI. Note that the images of the hidden B-52's did rate higher the the images of the KC-135's that were in clear view.

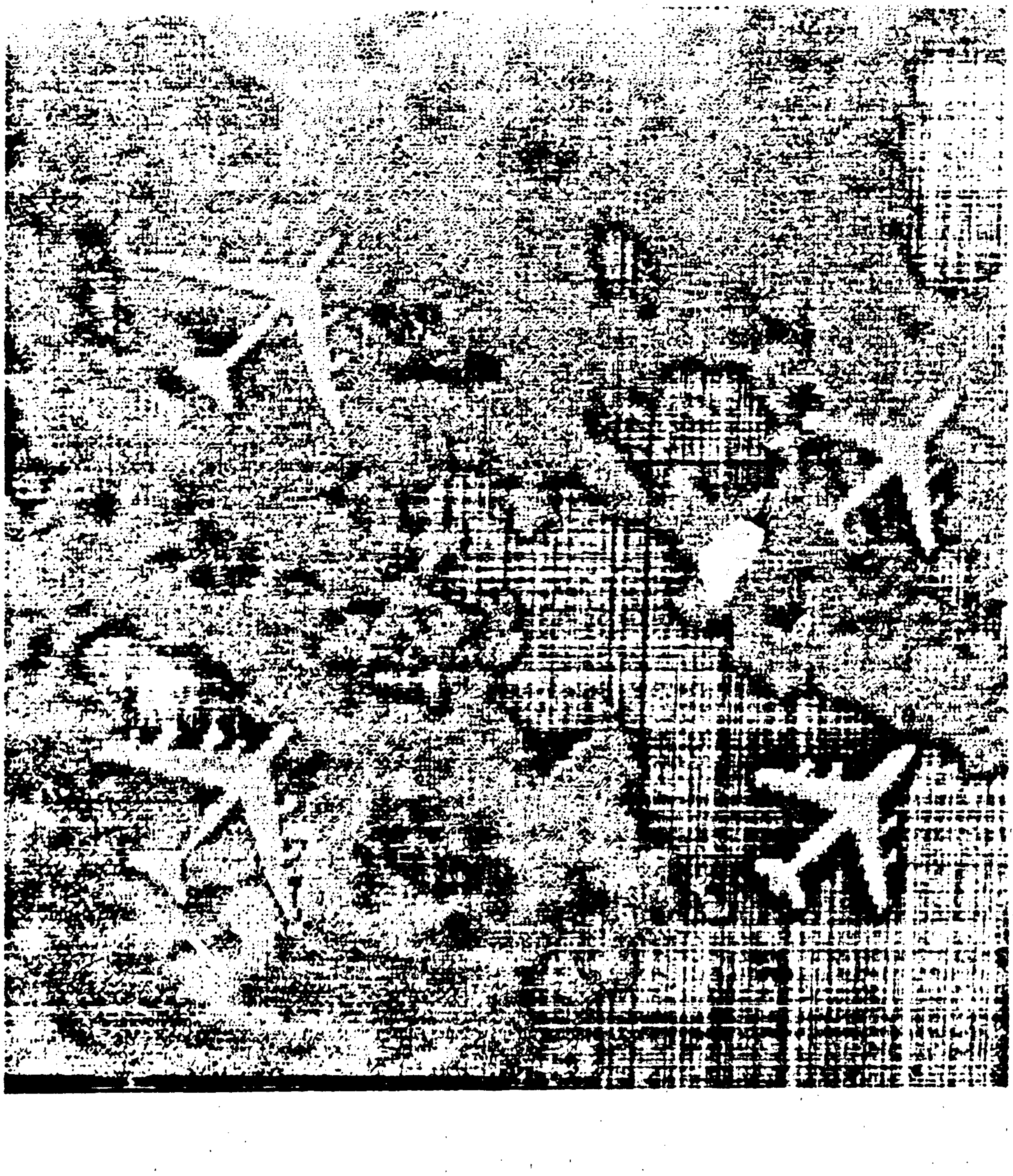


Figure E.20 Input Image of Test Case VI.

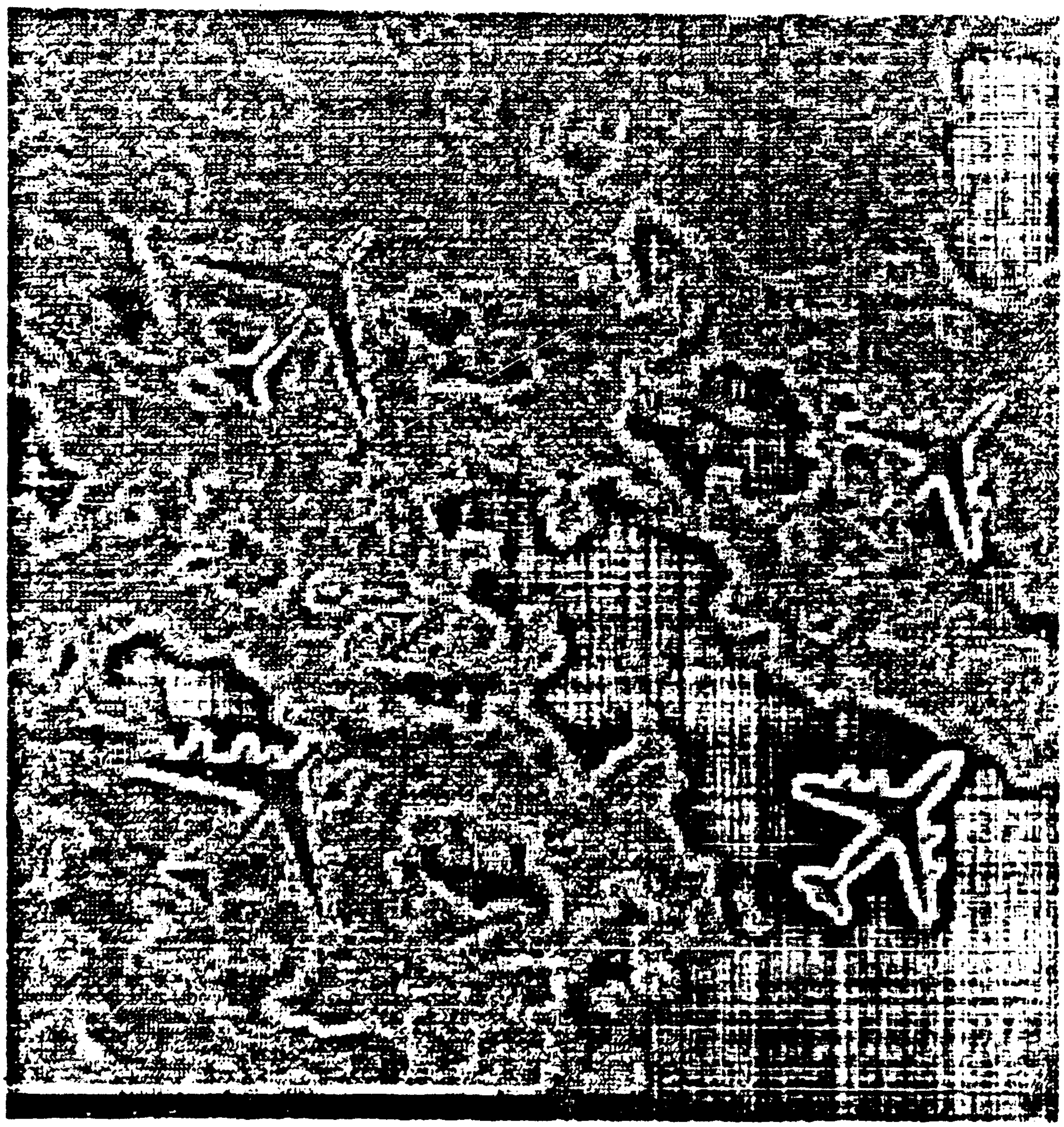


Figure E.21 Edge Extraction of Figure E.20.

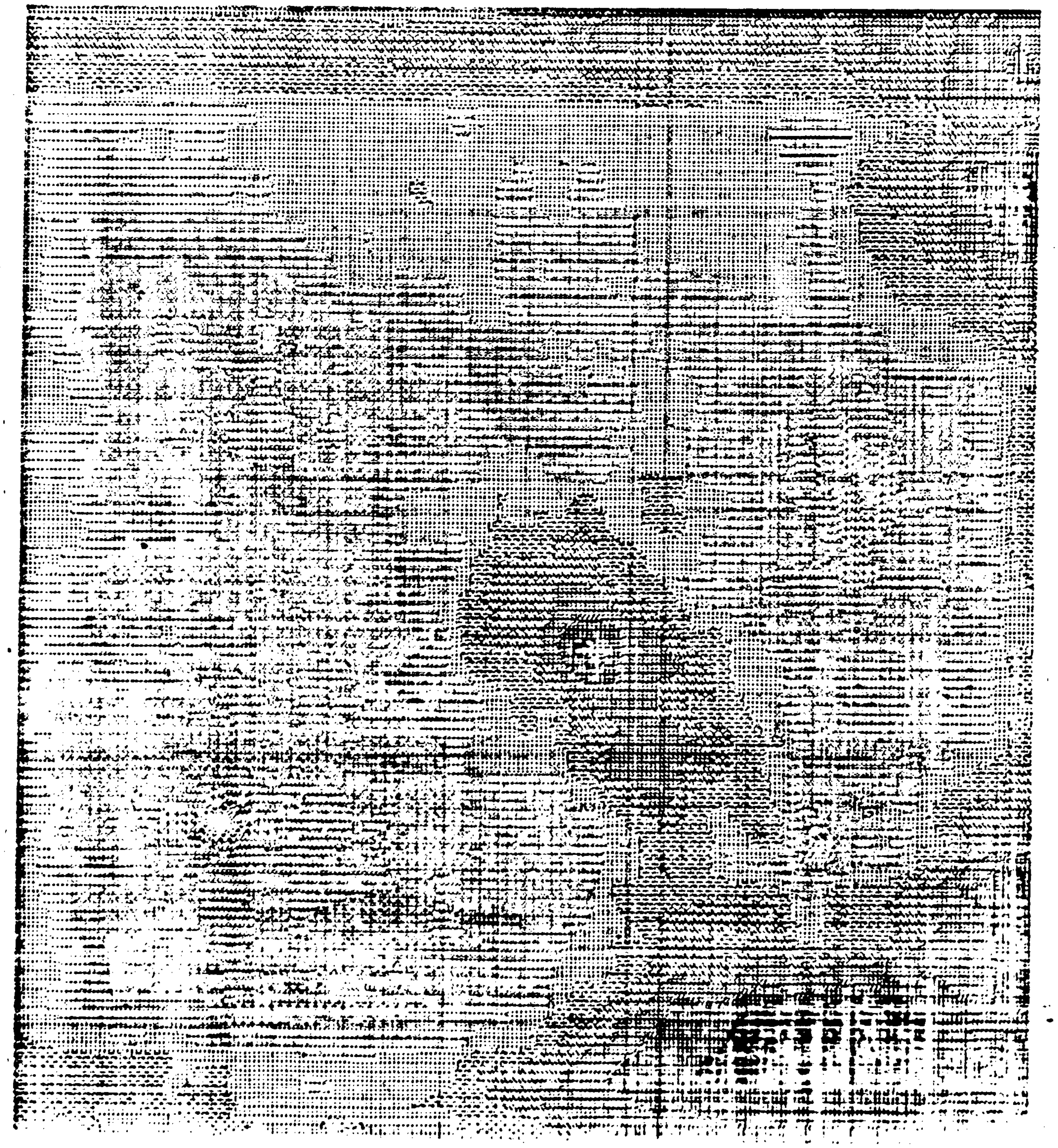


Figure E.22 First Pass Correlation Output of Test Case VI.

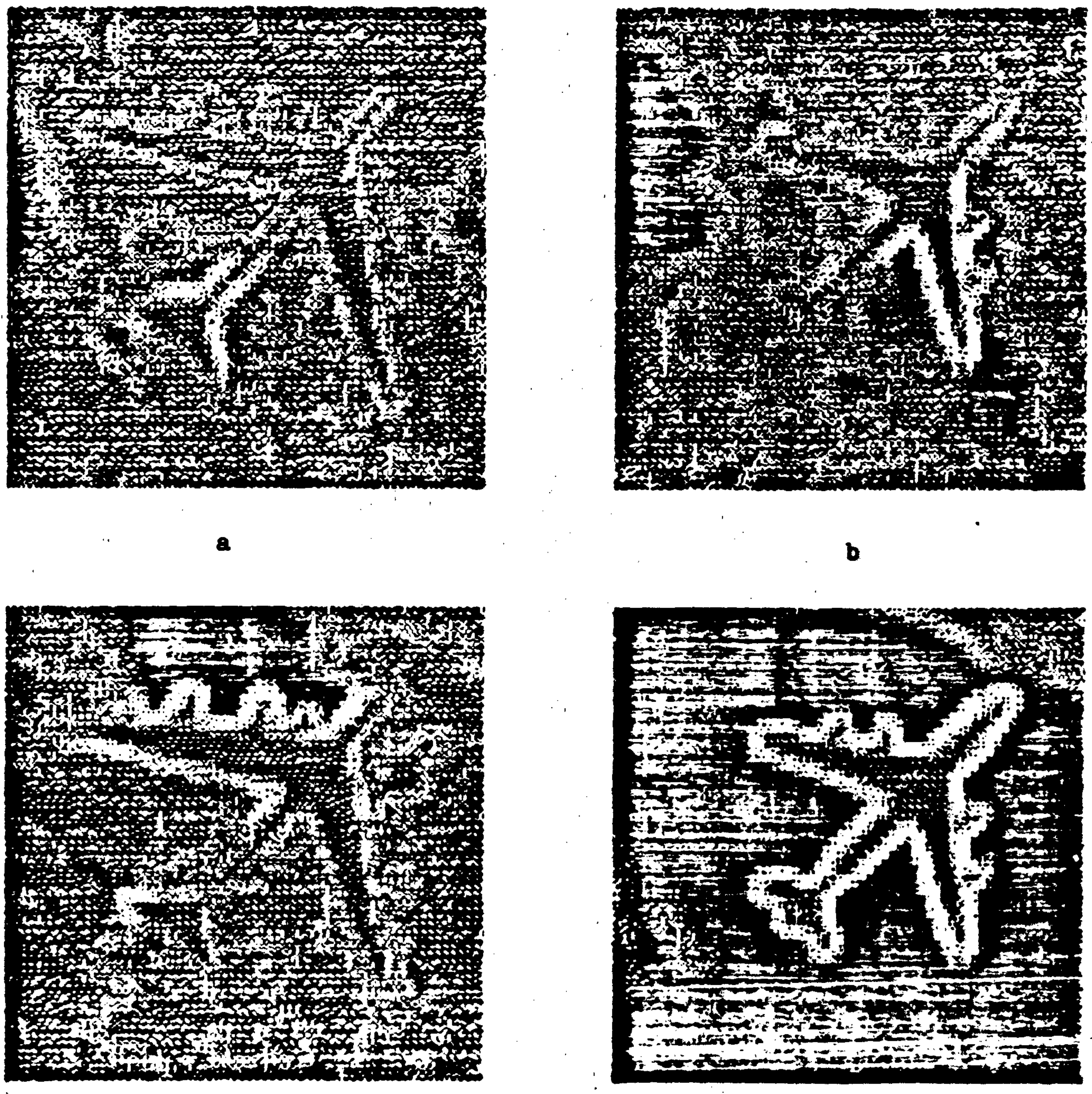
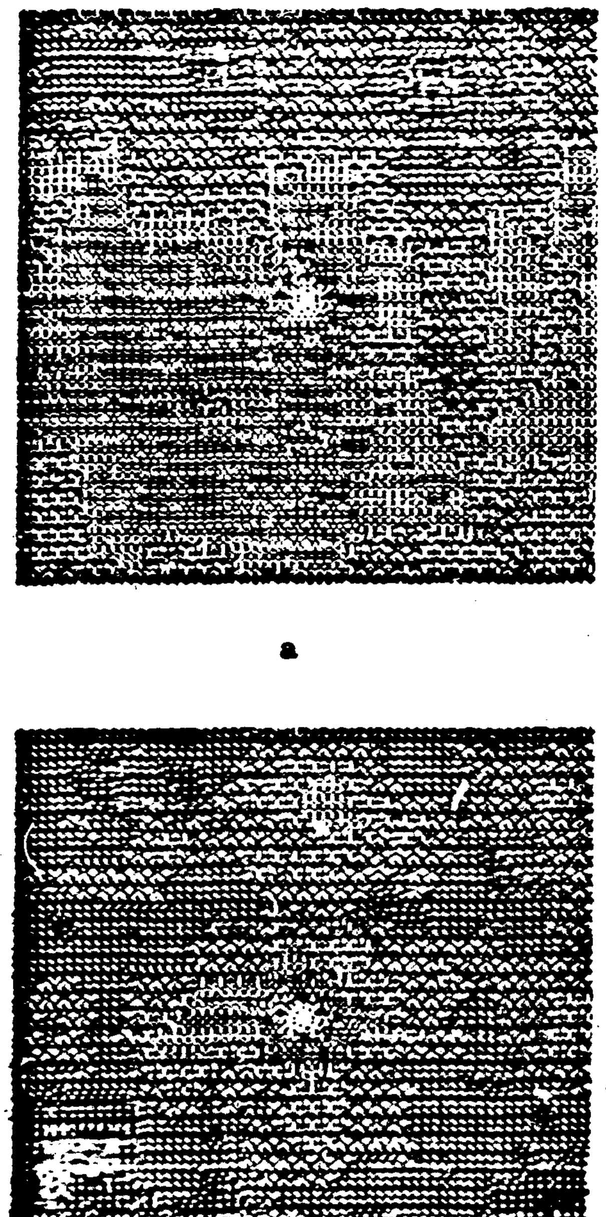


Figure E.23 Small Files Corresponding to Peaks in Figure E.22. (No reduction of PIX output)


a

b

c

d

Figure E.24 Second Pass Correlation Output of Test Case VI.
(No reduction of PIX output)

Table VI

Evaluation of Test Case VI

```
                    EVALUATION RESULTS
               SCENE FILENAME:   FIGURE E.24
FILE NAME          SCORE1       SCORE2     TOTAL SCORE      RANK
---------          ------       ------     -----------      ----
FIGE24A.VD        .977903      .873436       .854136          9
FIGE24B.VD        .977903      .655505       .641020          7
FIGE24C.VD        .977903      .980772       .959100         10
FIGE24D.VD        .977903      .720717       .704791          8
```

Test Case VII

The final test case included in this thesis is of an image that does not contain any targets. This is the cluttered cloud background, used in the two previous test cases, shown in Figure E.25. The edge extraction of the scene is shown in Figure E.26.

The edged target template used in this test case was the same B-52 template in Figure E.16. The first pass correlation output is shown in Figure E.27 The small potential targets, corresponding to the weak correlation peaks, are shown in Figure E.28. Four peaks were chosen as being locations of possible targets. The second pass, energy normalized correlations are shown in Figure E.29. The evaluation of these four recorrelations is given in Table VI. Note that two areas of noise ranked higher than the other two areas of noise due to their isolated energy distribution in the center of the scene which matched with the central energy of the aircraft template.

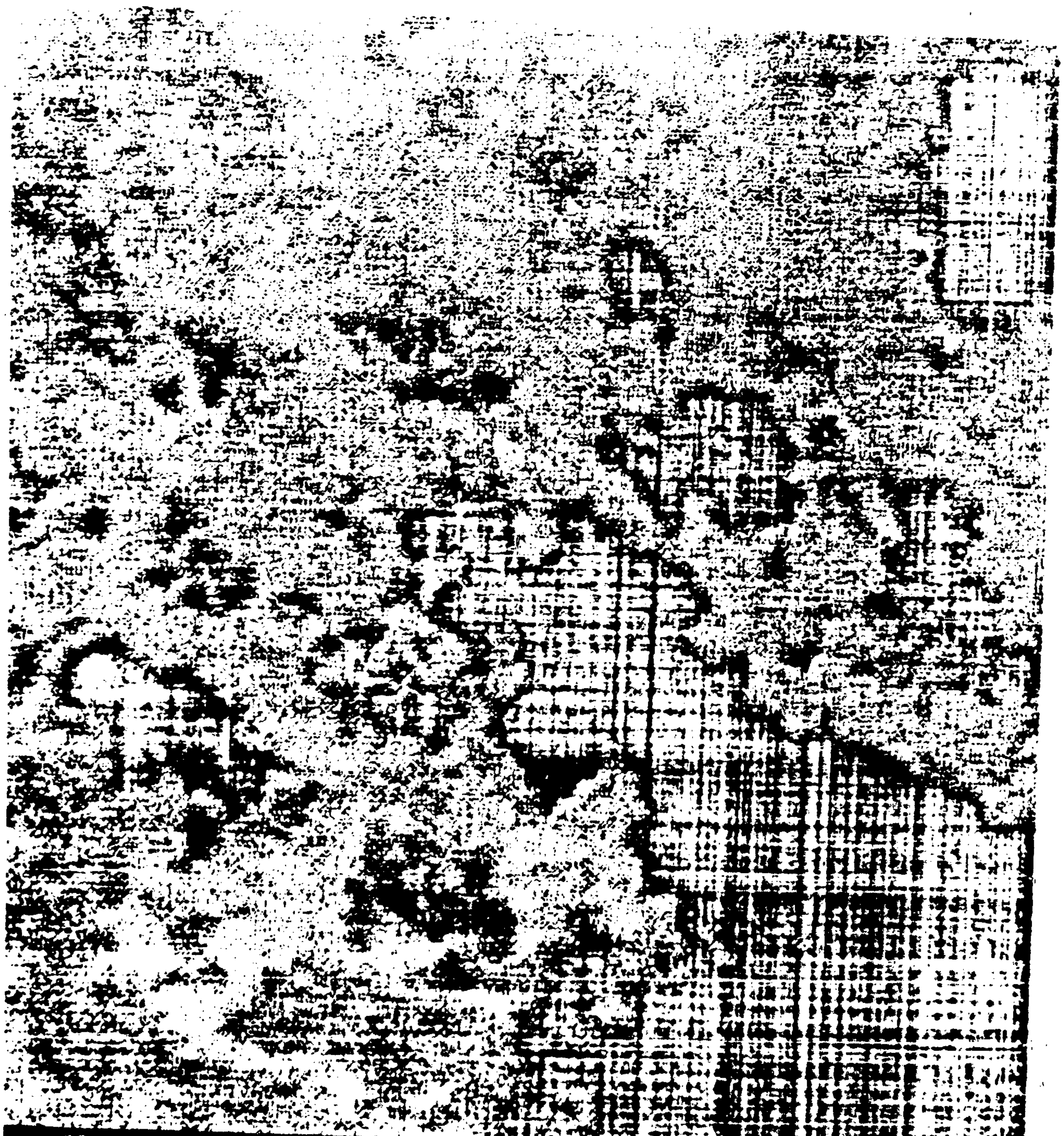


Figure E.25 Input Image of Test Case VII.

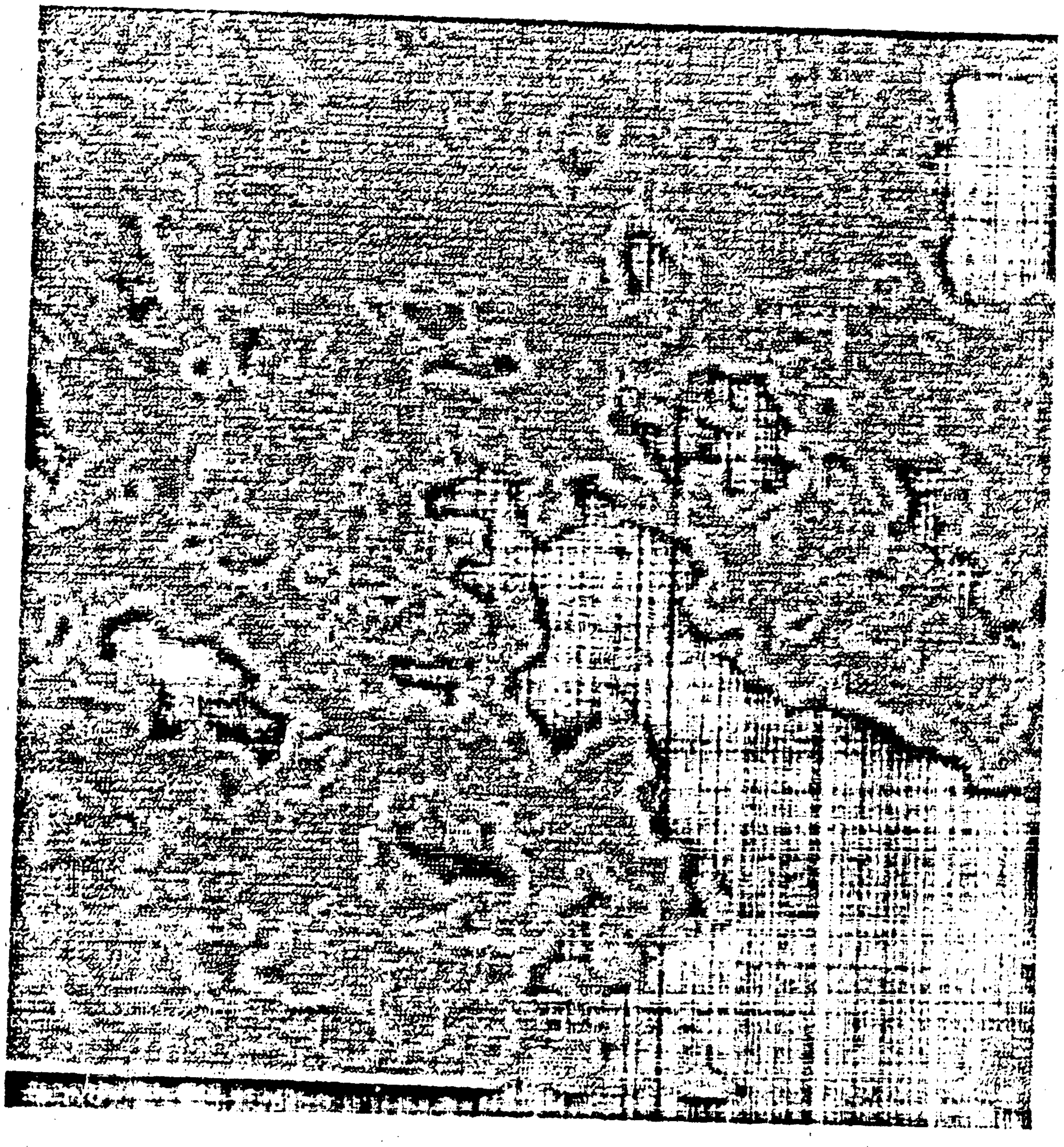


Figure E.26 Edge Extraction of Figure E.25.

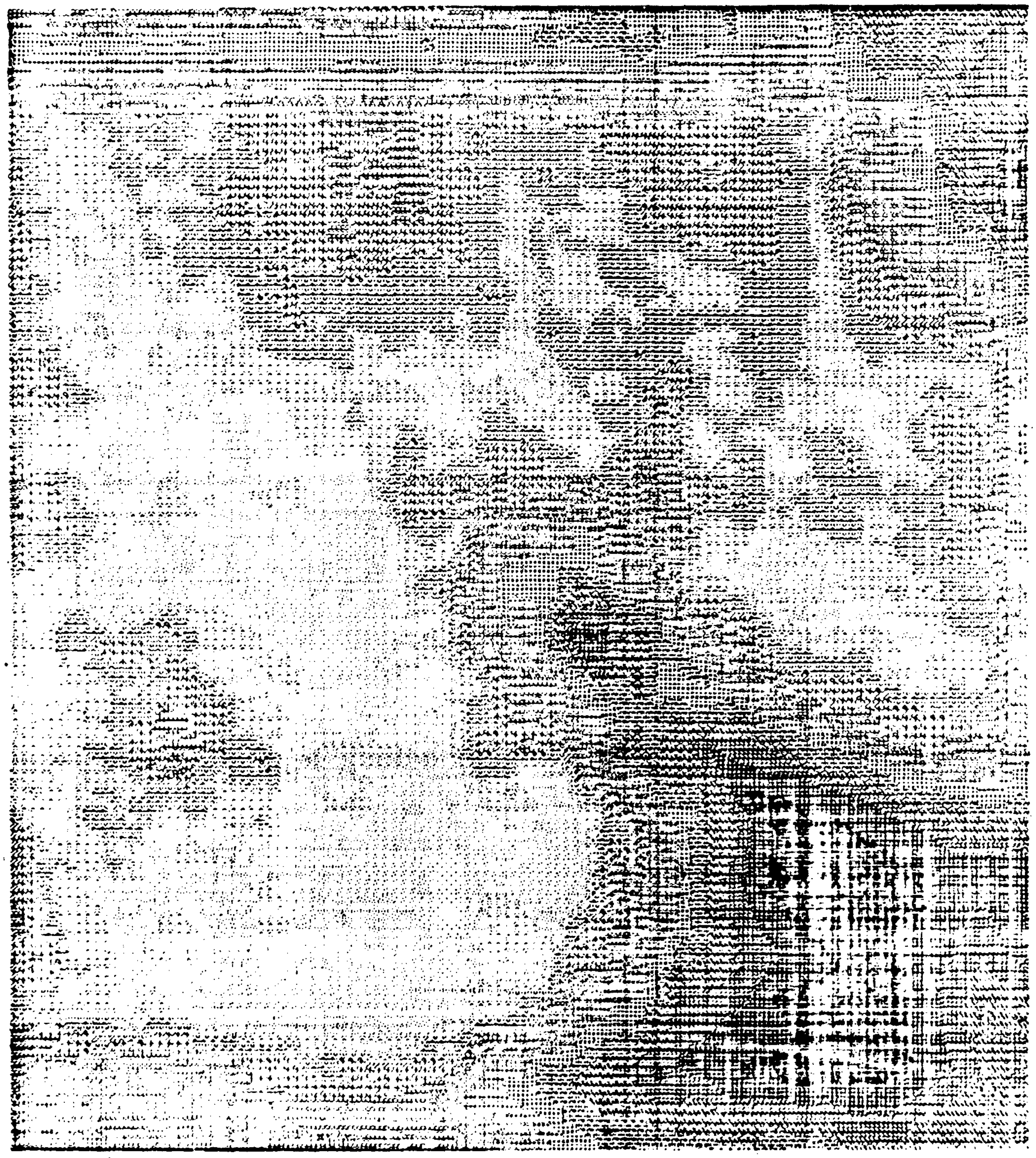
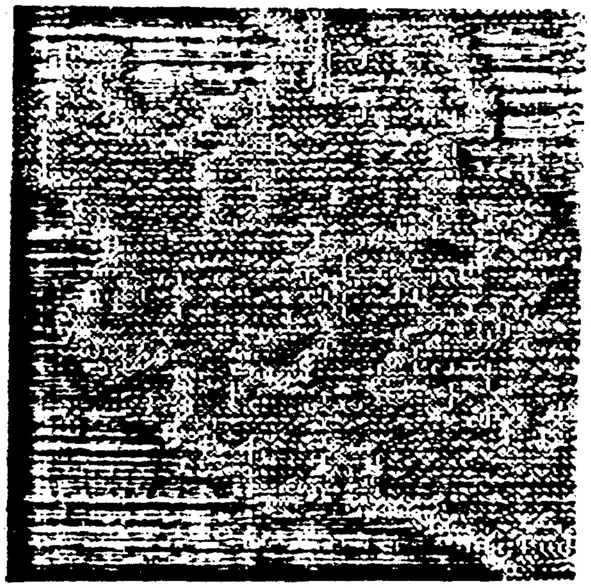
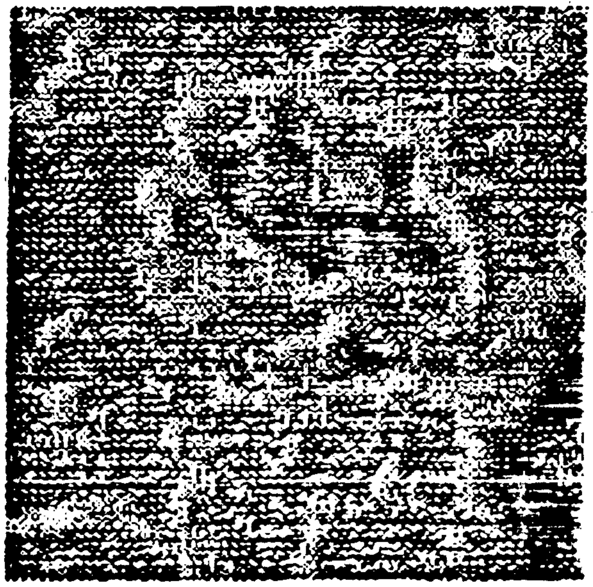


Figure E.27 First Pass Correlation Output of Test Case VII.

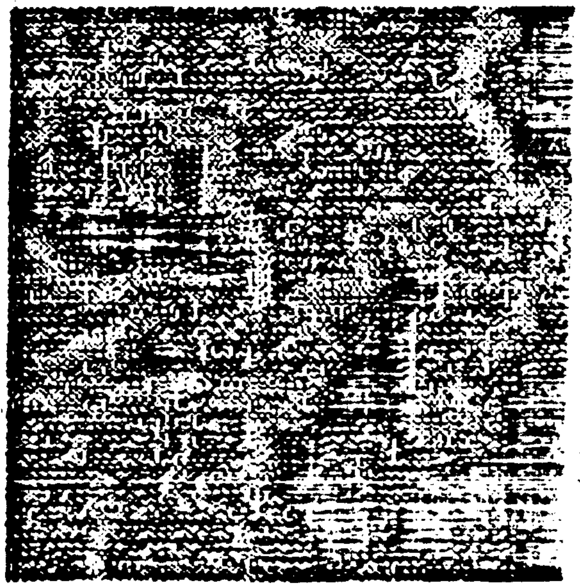
a



b



c



d

Figure E.28 Small Files Corresponding to Peaks in Figure E.27. (No reduction of PIX output)

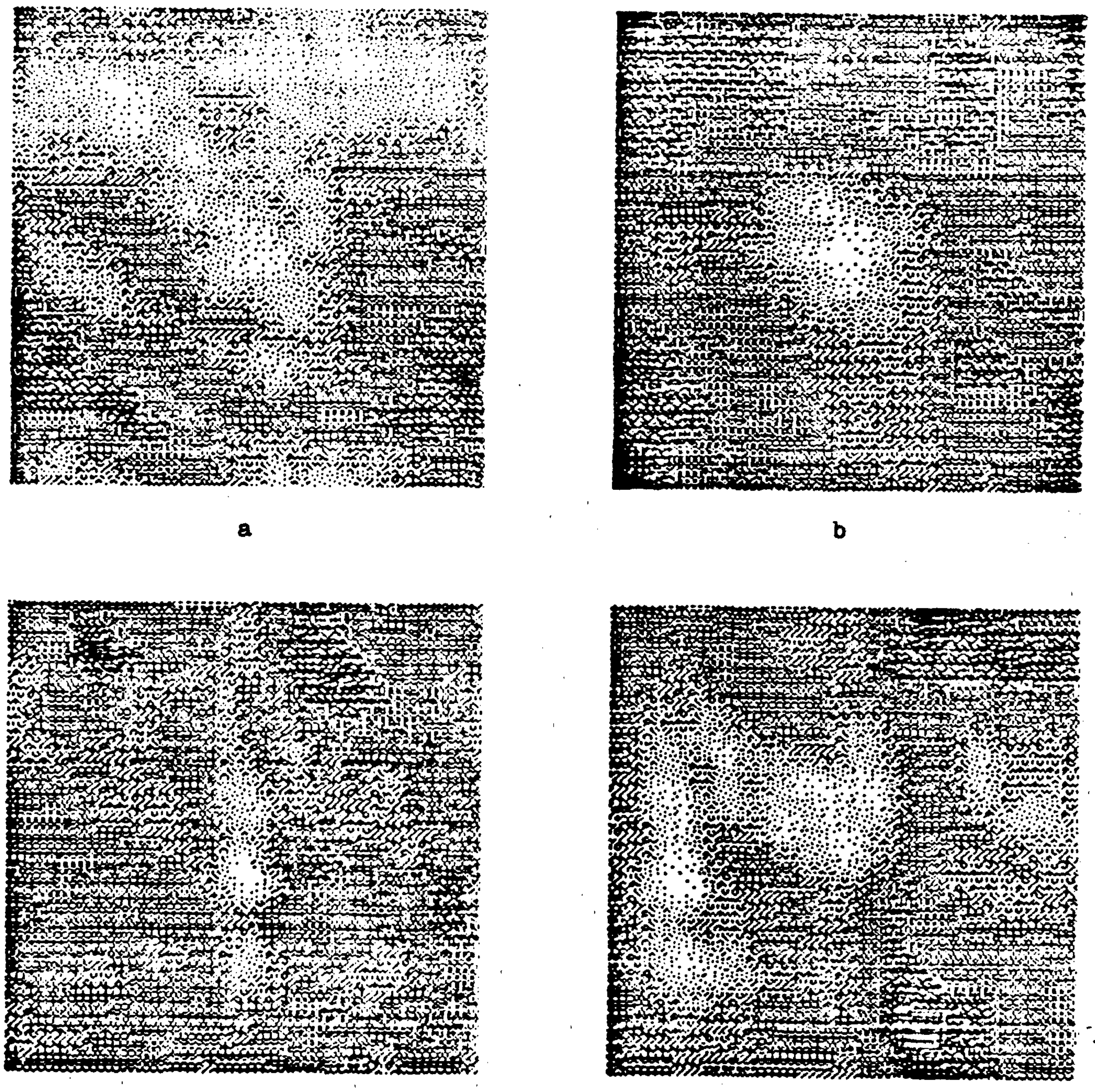


Figure E.29 Second Pass Correlation Output of Test Case VII. (No reduction of PIX output)

Table VII

Evaluation of Test Case VII

EVALUATION RESULTS
SCENE FILENAME: FIGURE E.29

FILE NAME	SCORE1	SCORE2	TOTAL SCORE	RANK
FIGE29A.VD	.125838	.416693	.052436	1
FIGE29B.VD	.950589	.560848	.533136	6
FIGE29C.VD	.858510	.649143	.557295	6
FIGE29D.VD	.276173	.457362	.126311	2

VITA

Richard L. Mills was born 3 May 1961 in Viborg, South Dakota. He graduated from high school in Lennox, South Dakota, in 1979. He then attended South Dakota State University on a 4-year AFROTC scholarship. He graduated with high honors recieving the degree of Bachelor of Science with a major in Electrical Engineering and a minor in Mathematics in 198 . At that time he was commissioned in the United States Air Force through the ROTC program.

After graduation, his first active duty assignment was to the Air Force Institute of Technology in May 1983.

He is married to the former Elizabeth Eppel.

Permanent address: 300 N. Main Street
Lennox, South Dakota 57039

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved forpublic relea[illegible]; Distribution unlimited
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) AFIT/GEO/ENG/84D-3	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION School of Engineering — 6b. OFFICE SYMBOL (If applicable) AFIT/ENG	7a. NAME OF MONITORING ORGANIZATION
6c. ADDRESS (City, State and ZIP Code) Air Force Institute of Technology Wright-Patterson AFB, Ohio 45433	7b. ADDRESS (City, State and ZIP Code)
8a. NAME OF FUNDING/SPONSORING ORGANIZATION — 8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (City, State and ZIP Code)	10. SOURCE OF FUNDING NOS.

PROGRAM ELEMENT NO.	PROJECT NO.	TASK NO.	WORK UNIT NO.

11. TITLE (Include Security Classification)
See block #19

12. PERSONAL AUTHOR(S)
Richard L. Mills, B.S.E.E., 2Lt USAF

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Yr., Mo., Day)	15. PAGE COUNT
MS Thesis	FROM ______ TO ______	1984 December	182

16. SUPPLEMENTARY NOTATION
[illegible]

[illegible] E. WOLAVER 25 Feb 85
Dean for Research and Professional Development
[illegible] Technology (ATC)
Air Force [illegible]

17. COSATI CODES

FIELD	GROUP	SUB. GR.
17	08	
06	04	

18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
Pattern Recognition; Target Detection; Cross-correlation; Normalizing.

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

SCENE ANALYSIS USING RECURSIVE FREQUENCY DOMAIN CORRELATION WITH ENERGY NORMALIZATION.

This thesis describes a scene analysis algorithm which locates targets in a noisy background. Preprocessing is used to detect the edges of objects in a digitized scene. The edges extracted from the scene is then cross-correlated with a template of the target to be found.

Cross-correlation is done by using complex-conjugate multiplication in the frequency domain. Areas of high correlation are recorrelated using smaller images which can be processed faster. The potential targets are energy normalized with respect to the template in order to eliminate false correlation with noise.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT [✓ UNCLASSIFIED/UNLIMITED] ☐ SAME AS RPT. ☐ DTIC USERS ☐	21. ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED
22a. NAME OF RESPONSIBLE INDIVIDUAL Dr. M. Kabrisky	22b. TELEPHONE NUMBER (Include Area Code) 513-255-5276 — 22c. OFFICE SYMBOL AFIT/ENG

DD FORM 1473, 83 APR EDITION OF 1 JAN 73 IS OBSOLETE.

SECURITY CLASSIFICATION OF THIS PAGE